COMBAT RATION ADVANCED MANUFACTURING TECHNOLOGY DEMONSTRATION (CRAMTD)

"Feasibility of Robotics and Machine Vision in Military Combat Ration Inspection"

Short Term Project (STP)#11

FINAL TECHNICAL REPORT
Results and Accomplishments (September 1991 through February 1993)
Report No. CRAMTD STP #11 - FTR 5.0
CDRL Sequence A004
June 1994

CRAMTD CONTRACT NO. DLA900-88-D-0383
CLIN 0004

Sponsored by:
DEFENSE LOGISTICS AGENCY
Cameron Station
Alexandria, VA 22304-6145

Contractor:
Rutgers, The State University of New Jersey
THE CENTER FOR ADVANCED FOOD TECHNOLOGY*
Cook College
N.J. Agricultural Experiment Station
New Brunswick, New Jersey 08903

Principal Investigators:
Stanley M. Dunn, Grigore C. Burdea and Kuan-Chong Ting

Dr. John F. Coburn
Program Director

TEL: 908-445-6132
FAX: 908-445-6145

*A New Jersey Commission on Science and Technology Center

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1994	Final SEP 1991 - FEB 1993

4. TITLE AND SUBTITLE
Feasibility of Robotics and Machine Vision in Military Combat Ration Inspection (Short Term Project - STP#11).

5. FUNDING NUMBERS
C-DLA900-88D-0383
PE-7811s
PR-88003

6. AUTHOR(S)
Stanley M. Dunn, Grigore C. Burdea and Kuan-Chong Ting

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rutgers, The State University of New Jersey
The Center for Advanced Food Technology Cook College
NJ Agricultural Experiment Station
New Brunswick, NJ 08903

8. PERFORMING ORGANIZATION REPORT NUMBER
FTR 5.0

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Logistics Agency
Cameron Station
Alexandria, VA 22304-6100

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Currently most quality control processes in a food product manufacturing plant are performed by human inspectors. In a highly automated plant this inspection is one of the most labor intensive operations. The use of machine vision techniques has just begun to gain acceptance in the food industry having been limited by special characteristics not present in other applications. This report describes a unique machine vision system for inspecting MRE pouches that is based on characteristics of the incident light, the surface imaged and their interaction. A prototype was constructed and the experiments and verification is described. The defect detection performance established technical feasibility. Information from an existing MRE Plant was used to create a numerical simulation model which was then adjusted to match the average performance of two other MRE producers. An interactive graphics simulation was created which allows the user to "fly" through the graphics model to visualize its 3D aspects. Based on the simulations and an economic analysis, the authors found that fixed transport automation is more economically feasible than a flexible automation system. It was concluded that automatic inspection of MRE pouches is both technically and economically feasible.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Machine Vision, Robotics, Product Inspection, Inspection Simulation	130
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Contents

1 Introduction

1.1 Introduction and Objective

In the last forty years, robotics has found widespread use in many industrial areas. Industrial robots have been used since the late 60's in manufacturing for spot and arc welding, machine loading and unloading, palletizing, spray painting etc. Recently more advanced techniques have allowed robotics to enter more skill-demanding fields such as parts assembly [43, 6, 36], quality control processes [72], medicine [46, 63, 21, 4, 22] and aero-space applications [51].

Some of the technological advances that have enhanced robotics are machine vision [62], tactile and force sensors [24, 12, 11], sophisticated control strategies [39, 52, 53, 59], high-speed, high-precision actuators [1, 55, 57, 50], dextrous end-effectors [20, 67, 35, 38, 12, 11], telerobotics [10], etc. As these technologies mature, robot manipulators will be able to perform most of the labor-intensive industrial tasks.

Although, there exist a large number of available advanced robotics technologies, there are still many areas of the industry where the application of robotics is uncommon. The food industry is one such examples [40]. However, initial research in automated food-handling has proved that robotics has a potential impact in three main fields [16]: food production [33, 64, 26], food processing [42, 41, 54, 19] and food services [18].

Domestic thermostabilized food packaging companies, including the DLA (Defense Logistic Agency) suppliers of combat ration MRE (Meal Ready to Eat) are not using available advanced manufacturing technology. The manufacturing of combat rations is still very labor intensive. The reliance on labor intensive efforts severely impacts the ability of the DLA to mobilize domestic resources to meet emergency needs in a cost and time effective way. In addition, these advanced technologies will contribute to the competitive posture of domestic industries.

The objective of this report is to study the feasibility of applying robotics and machine vision to the quality control of combat rations (in particular to the in–line inspection of MRE pouches). One of the goals of this research is to develop simulation and analysis tools that can help to evaluate the applicability of such advanced technologies. Another goal is to present an initial workable system that can be used as a "proof-of-concept" starting point for future developments.

What follows is a summary of the general aspects of robotics and machine vision in the food manufacturing industry. After this overview is a presentation of the research project that generated this work and the final report outline.

1.2 Robot Manipulation in the Food Industry

It is important at this point to clarify the difference between fixed-automation and flexible-automation (robotics). Many food industries are highly automated, using specialized machines to do automatic cutting, filling, packaging, palletizing, large-scale movement of materials, etc.. These fixed-automation machines replace efficiently human labor and are, in general, specifically designed to perform one given task.

Flexible-automation systems are also used to perform labor-intensive tasks, however, those robotics systems are flexible, programmable and with small modifications can easily perform a variety of task [58, 31, 3]. This flexibility is due mainly to the presence of programmable computers in the control chain and generalized mechanical design (robot arms like Unimate Puma, IBM Scara, Seiko RT-300, etc).

As opposed to fixed-automation systems, computer controlled systems can be programmed to make decisions based on external sensing [70]. This is one of the reasons why robots equipped with vision systems and sensor kernels are used in jobs were different manipulation actions are required for different environment states [9, 32, 47].

Fixed-automation mechanisms are used in two kind of processes. the ones that required little dexterity (labeling, filling, etc), and were it is possible to use a mechanical device that repeatedly performs the same task (sealing, tapping, etc). On the other hand, the robotics systems can handle more complex situations. First, because the computer controller permits a more sophisticated task planing. Second, because the mechanical design is generalized so that the device can adapt in real-time.

Large research efforts have been spent on the area of object grasping. One of the problems that makes difficult the application of robotics in food industry is the grasping and handling of objects with unknown position/orientation. The solution of this problem may require sophisticated detection devices. However some investigators have developed special techniques to simplify the problem. Goldberg [29, 28] discussed a Bayesian grasping method in which basic manipulation is used to orient an object whose initial orientation is unknown. The technique is applicable to hard objects with a polygonal shape (e.g. carton boxes or plastic containers).

Unfortunately not all objects have well defined shapes (piece of ham) making it difficult to apply these techniques. Nevertheless, several schemes have been developed to deal with these new situations. Lee et al. [44] presented a system using artificial intelligence and robot vision to determine the most optimal way to grasp and manipulate irregular shaped objects.

The stiffness of the object can invalidate many handling techniques. Very soft food products present a problem for a standard robot end-effector. Brett et al. [7] analyzed the problem of non-rigid product

handling. Among the non-rigid products are most of the food products. Tedford [64] analyzed the problem of end-effectors for soft fruit packing. Pneumatic and electro-mechanical grippers were discussed together with the control strategies. A comparison between human and machine prehension was presented. Tedford established that a reliable system must include a soft-touch feedback sensor.

Khodabandehloo [42] studied the specific problem of robotic handling and packing of poultry products. Due to shape irregularities of poultry products, special end-effectors were required. Some examples are contour adapting vacuum grippers, pneumatic rubber muscle fingers, moving belt tipped fingers, angle jaw grippers, etc. In addition to special grippers a machine vision system is required to detect position and orientation of the pieces to be grasped.

1.3 Machine Vision in the Food Industry

1.3.1 Robots and Vision Systems

From the examples cited in the previous section it is clear that a sophisticated handling device requires feedback information from the external environment. Sophisticated interaction needs knowledge about the external conditions. Such knowledge is used to take new control decisions or to correct errors.

Several sensors can be used to capture the conditions of the environment (ultrasonic proximity sensors, laser beams, magnetic sensors, etc). Machine vision (also called robot vision) is one of the techniques [37] used to register in real-time the conditions of the environment. In this case the images are processed to extract simple features like object size, shape, position and orientation. This information is then used by the robot controller to plan the task execution.

Vision systems permit the use of robot manipulators in application that demand great ability. For example Harrel et al. [33] developed a prototype robot for the citrus harvesting. The system uses robot vision and ultra sonic sensors to guide the special end-effector to the target.

Khodabandehloo [42] uses a machine vision system to deal with cut portions of poultry. The vision system is used to determine the orientation/position, the type of poultry pieces and for inspection. The machine vision system is integrated in a workcell together with the robot manipulator forming a complete robot workcell.

Machine vision can be used in conjunction with other feedback techniques to reach a high level of versatility. Purnell et al. [54] described the use of force feedback and robot vision for robot meat cutting. A stereo system is used to guide the end-effector cutter while the force feedback determines the appropriate force/torque to be applied at the robot's joints. Verghese et al. [69] presented a real time motion tracking of three-dimensional objects. The system can be used to track randomly oriented products that are being

transported by a conveyor.

1.3.2 The Quality Control Problem

One of the important applications of machine vision in the food industry is quality control (QC). Even though QC is one of the most labor demanding tasks, it can not be implemented by fixed-automation. Quality control processes is a task where decisions must be based upon sensor readings. Complex image processing is required to extract information from a digital image. This qualitative analysis can be made possible by the use of computers.

However, the use of machine vision techniques has just begun to gain acceptance in the food industry. The reason for this is that the quality control of food products demands special characteristics that are not present in existing applications [37]. Gagliardi et al. [27] discussed two of these characteristics, the color and texture measurements.

Currently most of the quality control processes (QCP) are performed by human inspectors. In a highly automated plant the QCP is one of the most labor intensive operations. This is because automatic machines can produce a large number of items per day under little supervision, however, all these items need to undertake one or more inspections.

This high volume of items to be inspected generates a new problem when using machine vision in QCP. In order to keep a reasonable productivity level, the inspection station must handle large amounts of items in a short time. Machine vision systems must acquire images, extract features and determine the quality in short periods of time (from 100ms to a few seconds).

There are several reasons for which it would be advisable to use robots in QCP. Robots can work longer shifts and their use con avoid contamination of product due to direct contact with human inspectors. However, economical analyses have shown that robotizing regular plants can be expensive, and the return on investment could take several years [65].

For this reason it is very important to be careful when studying robot applications in QCP. It is required to make use of low cost mechanisms to be able to determined whether or not robotizing is feasible. Computer simulations and small scale prototyping are two of the relatively inexpensive techniques for robot feasibility studies.

1.4 Computer Simulation for Feasibility Studies

Robotic systems may have a large impact on the productivity of industrial plants, for this reason, it is very important to study the feasibility of applications of such a technology before an implementation is

undertaken.

Several methods could be used to reach final conclusions. For instance, small pieces of the system can be prototyped so that experimentation is performed directly on them. However, because in many cases prototypes do not represent the complete system, it is difficult to obtain a general result. For instance, the performance of a assembly plant can not be completely determined if a prototype of a single line is built and analyzed. This is because an assembly line may be closely coupled with all the other lines. On the other hand, building a complete prototype is in many cases expensive.

Another way to study the performance of a system is by computer simulation. Numerical and graphics simulations represent two powerful tools for system feasibility analysis. The results obtained from a simulation model can then be extrapolated to the real system. Magnani [45] presented a complete discussion on modeling and simulation of an industrial robot. The aspects of sensor feedback for interaction with the environment are covered. In this way collision detection and real-time path planing are performed. Such results are proved to be completely applicable to the real robot.

The simulations can be used to predict malfunction under extreme conditions. Chang et al. [13] used a system model to evaluate the diagnosability of failure knowledge in manufacturing systems. This type of simulation permits to determine the weaknesses and limits of a system without performing dangerous experiments on the real system.

The process of building a computer simulation can be briefly outlined as follows [56]:

- A numerical model of the simulated system is built. This numerical model is made by selecting a set of variables that best describe the system and determines its behavior. The variables represent internal and external parameters of the system.
- Implement a computer program that represents these variables and their behavior.
- Then the program is used to simulate the system. This is done by varying the set of variables that represent external conditions while the program calculates the state of the internal variables.

In this way it is possible to monitor and analyze the system's behavior under different conditions.

A special case of computer simulation is graphics simulation. This type of simulation makes use of the real-time animation [68] to display a graphical representation of the model. In this case the variables that describe the system are the physical dimension and the spatial location. Troncy [66] developed a graphical interactive simulation that allows the user to program the robot by interacting with the graphic simulation.

1.5 Statement of Work

Combat Ration Advanced Manufacturing Technology Demonstration (CRAMTD) is a program of The Center for Advanced Food Technology (CAFT) at Rutgers The State University of New Jersey. CRAMTD is sponsored by the DoD-Defense Logistics Agency and a number of industrial partners. CAFT is one of the New Jersey Commission on Science and Technology Advanced Centers. One of the goals of CRAMTD is to demonstrate flexible automation technologies in the manufacturing of combat rations.

One of the short term projects of CRAMTD is STP11: "Feasibility of Application of machine vision and robotics in Packaged Food Manufacturing" (Principal Investigator Dr Stanley Dunn and the Co-Principal Investigators Dr. Grigore Burdea and Dr. Kuan-Chong Ting). The statement of work of STP11 is outlined as follows:

1. Scope: The scope of this project is to:

   (a) determine the specific tasks of the components required,

   (b) develop concepts for component design,

   (c) perform system integration by computer simulation,

   (d) determine system performance by computer simulation,

   (e) assess the cost effectiveness of the system,

   (f) recommend methodology for further development.

2. Technical Approach: The CRAMTD engineers will explore the feasibility of system integration including machine vision and robotics applications for MRE pouch inspection. In the case of machine vision, defective pouches will be collected from contractors, who will be asked to identify and classify the nature of the defects. These defective pouches will be subjected to several machine vision examination techniques, such as the use of colored light, the use of polarized light, the use of image processing algorithms and the use of range image techniques.

   The possible robotic workcell layouts and material handling scenarios will be studied in concert with the machine vision technique developed. The material manipulation actions required to accomplish the pouch inspection will be categorized into motions which need to be done by fixed automation and flexible automation concepts.

   Engineering parameters which affect the performance of the workcell will be identified. Computer modeling techniques will be used to develop component numerical models. The component models

will be integrated to form a system model. Design parameters will be varied and entered to the system's model to simulate the performance of the workcell under various conditions. The simulation process will facilitate the development of a workable system.

Engineering economic analyses will be performed on the workable system to compare the cost-effectiveness of different systems.

3. Program Schedule: This program will be conducted in a single phase consisting of 5 tasks and a Final Report as shown.

   (a) Review Current Design: Consists of a review with combat ration manufacturers and CRAMTD personnel. This review will identify the requirements for robot and vision workcells for MRE pouch inspection.

   (b) Identify Candidate Components: In this task, potential design strategies to satisfy the requirements will be established.

   (c) Develop Component and System Numerical Models: This task will involve the development of numerical models for workcell components and the entire system.

   (d) Computer Simulation to Identify Workable System Design: Using the models developed a series of design conditions will be systematically studied.

   (e) Economic Analysis: Under this task, evaluation of the system's cost-effectiveness shall be performed. The result shall be sufficient to recommend whether or not further commercial development is warranted.

   (f) Final Report.

The material developed in this report covers the topics included in the STP11 program schedule. Attention is focused on the computer workcell simulation, the robotic manipulation problem and the machine vision problem. The economical implications are considered in Section 6.

1.6 Report Outline

The first part of Section Two presents the problem of numerical simulation of the QCP and the relevant issues of the system modeling. The Section starts with a definition of the basic units that form a typical quality control plant. A model for defects and their detection during inspection is presented. The inspection time and inspection accuracy are discussed as a function of the item's quality.

The second part of Section Two describes the software developed for numerical simulation (the System Simulation Object Oriented Library "SOL"). A simple simulation program using C++ and SOL is explained. It shows how SOL facilitates the implementation of any QCP simulation by just mapping real entities and its interconnections to software objects.

The last part of Section Two presents the simulation of a real plant. The simulation is created with data gathered directly from three real food processing plants. An alternative simulation model using robotic inspector characteristics is developed as well. The two simulations are compared and conclusions on automatic vs. manual plants are obtained.

Section Three describes the interactive graphics simulator IGAS. This simulator is a complement to the numerical simulation. As opposed to quantitative characteristics (numerical values), IGAS permits the exploration of qualitative characteristics (physical shape and location) of QCP models Both, a human based and a robot-based workcell are simulated using IGAS. The user can interact with the graphics simulation by means of a track-ball device which allows the user to "fly" through the simulated workcell.

In Section Four the problem of pouch manipulation is presented. The pouch handling is separated into three zones. The characteristics and manipulation needs of each zone are analyzed. Two models that satisfy the handling requirements are presented. The models differ mainly by their level of automation. The first uses a commercial industrial robot while the other uses a custom designed manipulator. In both cases, the general design and control requirements are explained.

Section Five is a discussion of the machine vision technique developed to detect the defective pouches. The physical principles and a mathematical model for light reflection that is the fundamental basis for the inspection system is presented. The limitations and capacities of this new approach to inspection are explored, concluding with a model of how these fundamental physical principles can be used to inspect MRE pouches automatically. In addition, several alternative techniques for defect detection are suggested.

Section Six is a summary of our economic analysis of the two proposed automation systems. The section is divided into two parts: The first part is an overview of the computer based economic analysis system that was chosen for this phase. The second part of section 6 is a summary of the results for workcell versus human costs, equivalent capacity, feasibility analysis and parametric analysis.

Section Seven presents the conclusion of this research. A selection of the C++ code used to create the simulation is presented in Appendix A. The same appendix includes the code for the classes definitions of the SOL library.

1.7 Conclusions and Recommendations

In summary, this report describes a machine vision system for inspecting MRE pouches that is based on fundamental physical characteristics of the incident radiation, the surfaces imaged and their interaction. The optimizations described herein have not been used before in the food inspection industry.

Based on this theoretical model, a proposed machine vision paradigm was established and a prototype embodiment of this paradigm was constructed. The prototype built for experimentation and verification is described in this report. The results of defect detection experiments showing its diagnostic performance establishes technical feasibility of the imaging paradigm.

Two approaches for robotic transport through the inspection workcell were considered. The first is a fixed or hard automation system and the second is a flexible automation system. Both of these approaches were simulated and the result are contained herein. Based on the simulations and the economic analysis of each, we find that the fixed or hard automation system is the more economically feasible of the two approaches. In quantities, the system has a simple payback in under two years with a significantly high return on investment.

Thus, we conclude that automatic inspection of MRE pouches is both technically and economically feasible.

2 Numerical Simulation

2.1 Introduction

This section is divided into three parts. In the first part, the QCP is decomposed into several simple independent components. Each component is discussed in detail. Based on the components main characteristics generic numerical models are proposed.

In the second part, the System-simulation Object-oriented Library "SOL" is presented. SOL is a library of C++ classes (objects) used to build numerical simulations of QCP. Each one of the SOL's classes is a C++ implementation of the models of QCP's components. An illustrative example of a simulation of a single-line inspection workcell using SOL is discussed.

In the last part, a simulation for an existing plant is built. The results of the simulation are compared with real data collected from industry. Finally, the simulator is used to build a plant with the characteristics and parameters of a robotized plant. The topology and some controllable parameters are varied so that the robotized plant matches the performance of the human-operated plant. The two simulations are compared and conclusions are given.

2.2 System Model

Controlling the quality of products is, in general, a complex task which involves many steps. However, typical quality control systems can be decomposed into several simpler units that perform specific tasks. Such tasks can be easily identified, and computer models can be built for them.

A description of the basic components that form a generic quality control inspection workcell will now be presented. At this level of the model no distinctions are made of whether or not some components are implemented by human operators or automatic machines. This generality permits the model to be extrapolated from one extreme to the other by just changing some performance characteristics.

This extrapolation from human-operator to robot-operator will validate the feasibility of the application of robotic technology in a quality control process.

2.2.1 Workcell Component Modeling

Here, the basic components of a generic quality control process are defined. The basic components will be kept as general and simple as possible, so that, numerical models can be easily developed. The idea behind this simplicity is not to ignore complexity, but on the contrary, to have the building block required to achieve the implementation of more complex models. Some of the components' characteristics will be defined in

terms of some general variables or data structures that can be implemented in any computer language. The reason for this descriptive selection is to keep the model aimed towards a numerical simulation.

Item: The parts whose quality is to be controlled will be called *items*. An *item* can be of any nature and is considered to be the minimum unit of production to be manipulated. The main characteristic that is relevant for simulation purposes is the *item*'s quality that can be represented by a real value as discussed later. In addition to quality characteristics, *items* carry an identification data structure that could contain information such as serial number, date of manufacturing, contents, inspection results, etc.

Source: At the beginning of every quality control workcell there are *source* units. *Source* units emit all the items that flow through the inspection process. In simulation, *sources* are in charge of creating the data for the items emitted, including their defects. In a real situation, a *source* could be one of many different things: containers, warehouses, process emitting items, etc. However, for simulation reasons these distinct natures are irrelevant. What is important during simulation when dealing with *source* units are their output rate –measured in *items/sec*– and their quality distribution.

The output rate of any *source* unit is in general a normal distributed random variable –characterized by its mean value and dispersion–. The quality distribution says what is the probability of an emitted item to have a given quality. A broken *source* emits a larger number of defective items than a regular one. The functionality condition of a *source* component can be modeled by a uniform distributed variable representing the reliability of the given *source*. A threshold value is selected so that when the reliability variable is below (or over) this value the *source* is considered to be broken.

Buffer: During the path followed by the items in a QCP, the items may have to be temporarily stored or stacked in special places. This is because some parts of the process could take longer than others and the items coming from a previous faster process have to be temporarily accumulated. These places are called *buffer* units and are characterized by their capacity measured in items. For the simulation models in discussion, *buffers* are considered to be passive elements. That is, the only way to access them is through transfer units (discussed later), which access the *buffer* by putting in or taking out items. In other words, *buffers* by themselves are not able to get or send items from or to other units.

Inspector: The process of controlling the quality of the items is called "inspection". The unit or element in charge of inspection is the *inspector* unit. *Inspectors* look for a specific set of defects that has been established, and reject or mark items that do not satisfy quality standards. The time required for an *inspector* to inspect completely an item is called *inspection time*. The *success rate* is the percentage of times that an *inspector* is able to assert the correct quality of the items. The *inspection time* and the *success rate* characterize completely the *inspector*'s performance. Those variables that depend on the

quality characteristics of the particular item being inspected are discussed later.

There are several special classes of *inspectors*. Some *inspectors* have the special function of tabulating data from rejected items or supervising the work of a regular *inspector*. The characterization of special *inspectors* could require special variables different from the *inspection time* or *success rate*. Those cases are discussed independently and are considered to be exceptions. All *inspectors* access the items via a transportation unit that brings/takes the items from/to the inspector.

Conveyor: The transportation of the item through the process is accomplished by means of units: the *conveyors* and the *transfers*. *Conveyors* carry items for a fixed path at a constant or variable speed. In general, the *conveyor*'s path is such that *inspectors* or other units can take items and put them back in the *conveyor* after some processes have been performed (marking, registering, inspecting, etc). There are two parameters that characterize *conveyors*, the *capacity* and the *transport time*. The *capacity* is considered to be fixed, however, the *transport time* is a function of the *conveyor*'s speed and could change during the process. Some *inspectors* and other special units have the ability of controlling the speed of the *conveyor* if required. *Conveyors* could be used to move *items* between units, but this case occurs only when no special manipulation at the front end of the *conveyor* is required, for example: stacking, palletizing, depalletizing, etc.

Transfer: *Transfer* units are in charge of moving items between units. *Transfer* units can execute simple organizational tasks such as stacking, palletizing, depalletizing before or after the items are moved. Even activities like air-drying before palletizing (or depalletizing) can be assigned to *transfer* units. For simulation, the spatial organization of items into containers or stacks is not relevant, however, what is taken into account is the time required for such a manipulation. That is why the main characteristic of a *transfer* unit is the time required to complete the transfer process and be ready to receive or to take new items. This time is called *transfer time*. The capacity of a *transfer* unit (*transfer capacity*) is also important as they could move more than one item at a time.

Another important task performed by *transfer* units is selecting the path of inspected items. As mentioned before, *inspectors* mark or reject *items* with low quality. When the *items* have been just marked the *transfer* unit that follows the inspection will be in charge of discarding the bad item. For simulation, *transfer* units have an extra feature. Because they are used to interface two or more units, all the time delay can be easily lumped into the *transfer* units.

Accepted & Discarded: Once an item has gone through the inspection process it may be classified as either accepted or discarded. The final destination of the accepted items is called the *accepted* unit. In real cases these are large warehouses or large buffers that store items before they go through a new process

(if required). For the simulation, the *accepted* units are just unlimited capacity buffers that perform some statistical analyses using the accumulated data of the accepted items. The counterpart of the accepted unit is the *discarded* unit that receives the bad *items* and performs equivalent statistical analyses.

Processes: Manufacturing products is, in general, a multi-stages process. In many cases produced items have to undertake quality control inspection between processes. In simple simulated cases *process* units can be considered just as delays in an item's path. However, a *process* could eventually change the *item*'s quality. Furthermore, new types of defects have to be added or deleted to the defect table after *items* have passed through some processing. If the complete quality control workcell is to be considered, *processes* have to be included during simulation. For the simulations discussed in this report, *processes* are considered to be large buffers with large delays.

2.2.2 Modeling Defects

In general defects can be of any nature, however, for specific products there exists a set of specific types of defects. In such cases, defects are well known and their characteristics are tabulated and established before any quality control is undertaken. *Inspectors* are trained (or programmed) to discover such defects and to be able to decide quickly and with high accuracy whether or not the quality of the *items* is adequate.

To model defects in simulation it is not required to deal with the defect's intrinsic nature. What is important is to know its statistical behavior. Defect probability distributions or defect rates are two statistical characteristics that can be obtained from experimental sampling or just from data coming out of real QCP plants. Standard simulation techniques are used in this modeling [56].

To characterize quantitatively the set of defects $D = \{d_1, d_2, ..., d_N\}$, a quality level q_i is assigned to every defect d_i . In this way, the overall quality of an item is given by the set of quality levels $Q = \{q_1, q_2, ..., q_N\}$ corresponding to the set of defects D . The defects considered here are assumed not to be correlated, hence the quality levels q_i *and* q_j ; $\forall\ q_i, q_j \in Q$ are independent.

The simplest model for quality levels q is to assume two (binary) values of quality, good q_{good} or bad q_{bad}, each one with a given probability. To generate a variable with such characteristics is just a matter of using a uniform distributed variable u and a threshold value $U_{threshold}$ (Figure 1a). The value of q is then defined as:

$$q = \begin{cases} q_{good} & if\ U < U_{threshold} \\ q_{bad} & if\ U \geq U_{threshold} \end{cases} \tag{1}$$

such a binary distribution is shown in Figure 1b.

However, for a more realistic item-inspector interaction (discussed in the next subsection) it is better

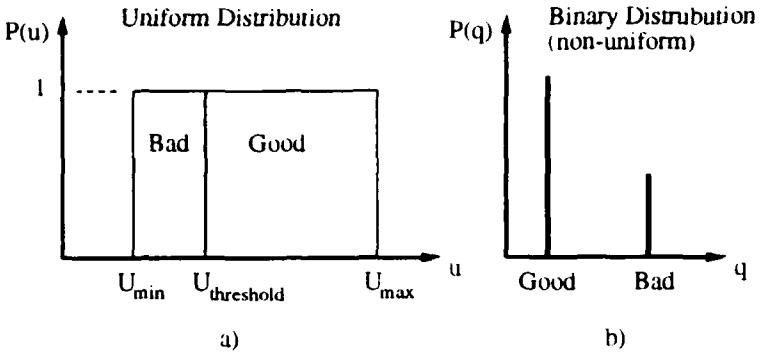


Figure 1: Binary non-uniform distributed variable

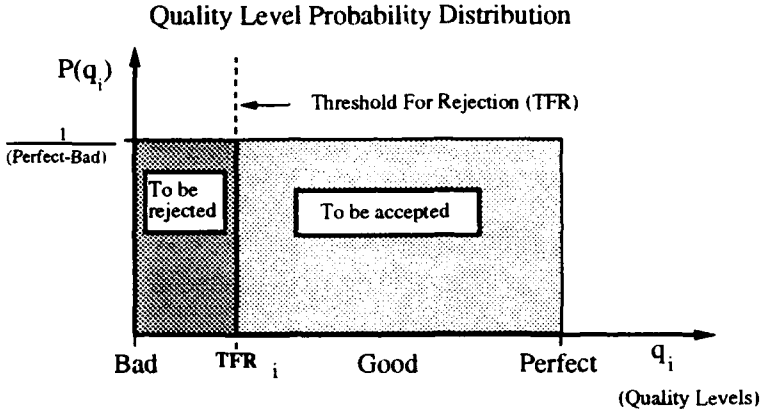


Figure 2: Quality distribution

to allow a larger range of quality levels. The following paragraphs presents a description of the quality modeling used for the simulations.

Figure 2 shows the probability distribution of the quality level q_i associated with defect i. It is just a uniform distribution on the interval $[Bad, Perfect]$. The TFR_i (Threshold For Rejection) level is the threshold level for quality control assigned to defect i. The quality control process can be described as follows: Items that have any of the quality levels $q_i; i = 1, 2, ..., N$ falling in the interval $[Bad, TFR_i]$ must be rejected. Hence, items with q_i in the interval $(TFR_i, Perfect]$ are considered to satisfy the quality standards and are accepted. (Note how the last interval has an open boundary in the value TRF_i to avoid conflicts).

For numerical simulations the interval $[Bad, Perfect]$ is normalized as the interval $[0, 1]$ hence

$0 \leq TFR \leq 1$. The $q_i;\ i = 1, 2, ..., N$ are generated by means of a uniform pseudo-random generator. The $TFR_i;\ i = 1, 2, 3..., N$ are fixed before the simulation. Their values are chosen so that $TFR_j * 100 = P_j$, where P_j is the percentage of defective items with defect j . During simulation, when an item m is to be issued from a source unit, the set of qualities Q_m associated to item i is created via a random generator. This N component vector is attached to the data structure of item m .

What follows is a description of the interaction of the items with the inspectors. It will be shown how the quality levels influence the inspection results qualitatively and quantitatively.

2.2.3 Modeling the Inspection Process

The emphasis will now shift to the components of the inspection process. The required definitions are introduced as well as the description of how the inspection is carried out. The two main aspects of the inspection performance are covered. The inspection time and the inspection accuracy. The techniques used here to correlate stochastic variables are described in [56].

The performance of any inspector could be affected by the quality level of the item being inspected. If the quality is so good that the item could be considered "perfect", the inspector does not have to spend extra time deciding or doing further inspection. If the quality is very bad, the inspector can reject very quickly the item. However if the item's quality is such that it is close to the threshold for rejection TFR, any inspector could spend extra time trying to determine the correct quality of the item. Furthermore, in this particular situation inspectors are most likely to misjudge. In other words, items with a quality level in the neighborhood of the threshold level TFR are critical for inspection. For these items the inspection time increases and the inspector accuracy decreases.

2.2.4 Inspection Time

The total inspection time of inspector k required to inspect one item is denoted as t_k^*, and is defined as:

$$t_k^* = \sum_i^N t_{ki}, \tag{2}$$

where t_{ki} is the time used by inspector k to inspect defect i alone. The sum goes over all the set of defects.

t_{ki} is modeled by means of a normally distributed bounded variable with probability $P(t_{ki})$ shown in Figure 3 and given by:

$$P(t_{ki}) = \alpha e^{-\frac{\overline{t_{ki}}^2}{2}} \tag{3}$$

for $t_{ki}^{min} < t_{ki} < t_{ki}^{max}$ and:

$$P(t_{ki}) = 0 \tag{4}$$

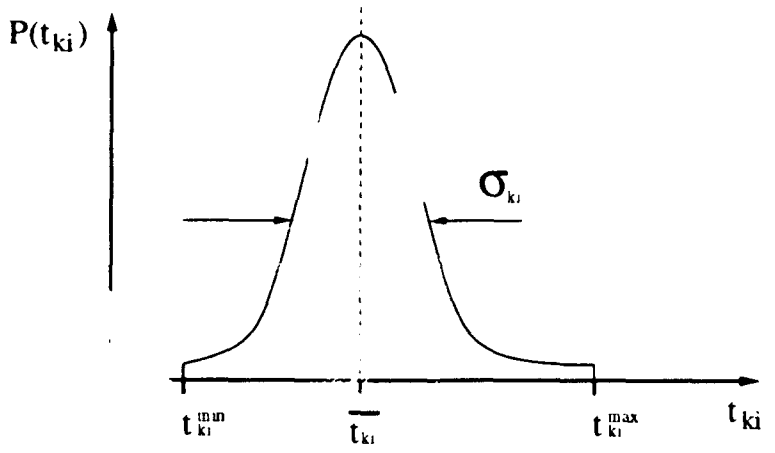


Figure 3: Inspection time distribution: Inspector k, defect i

for $t_{ki} < t_{ki}^{min}$ or $t_{ki} > t_{ki}^{max}$. The latter bounds the variable t_{ki} to avoid unrealistic values.

To implement the correlation between item quality and inspection time, the mean $\overline{t_{ki}}$ is made a function of the quality level q_i of the particular item being inspected. This relation is shown in Figure 4 and is given by:

$$\overline{t_{ki}} = \begin{cases} T_{ki}^{Bad} + \frac{q_i}{TFR_i} T_{ki}^{TFR} & \text{if } Bad \leq q_i \leq TFR_i, \\ T_{ki}^{TFR} - \frac{(q_i - TFR_i)}{1 - TFR_i}(T_{ki}^{TFR} - T_{ki}^{Perfect}) & \text{if } TFR_i < q_i \leq Perfect \end{cases} \tag{5}$$

The values $T_{ki}^{Bad}, T_{ki}^{TFR}$ and $T_{ki}^{Perfect}$; $i = 1, 2, ..., N$ are constant parameter of the inspector k. The maximum average inspection time $t_{ki} = T_{ki}^{TFR}$ occurs when the quality of the item q_i is equal to the threshold for rejection TFR_i.

2.2.5 Inspection Error

The inspection process is a stochastic process. No inspector is able to determine with total accuracy the real quality level q_i corresponding to defect i of a given item. Inspector k judges the quality q_i by doing an estimate $\hat{q_{ki}}$ which is compared against the TFR_i. Inspectors are trained or programmed to minimize the error $|q_i - \hat{q_{ki}}|$. However, as mentioned before, the estimation error could increase as the quality level approaches the critical value TFR_i.

A good simulated model is obtained when the estimate $\hat{q_{ki}}$ is taken as a normal distributed random

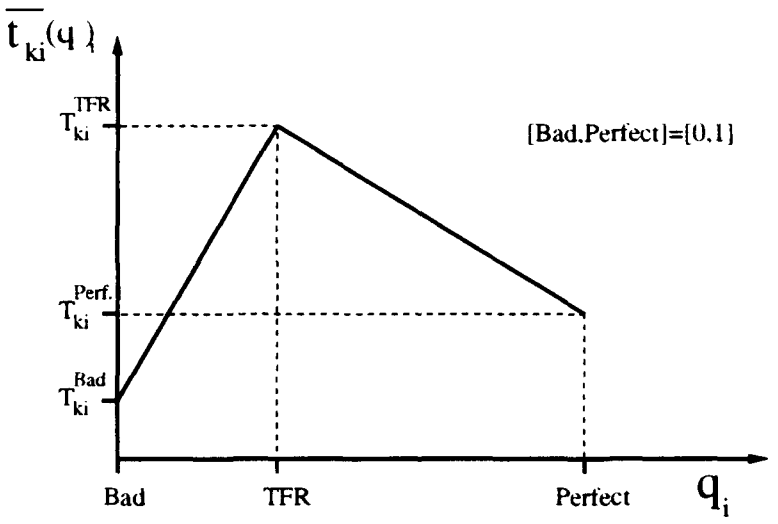


Figure 4: Average inspection time: Inspector k, defect i.

variable with mean $\tilde{q_{ki}}$ and standard deviation σ_{ki} . In this case the mean value $\tilde{q_{ki}}$ is the real quality level q_i ; $\tilde{q_{ki}} = q_i$. So the estimate is normally distributed around the real value.

To model the fact that the estimation error depends on the value of q_i the standard deviation σ_{ki} is made a function of q_i . A similar function to the one used for the mean inspection time is used here. This function is shown in Figure 5 and is given by:

$$\sigma_{ki} = \begin{cases} \sigma_{ki}^{Bad} + \frac{q_i}{TFR_i}\sigma_{ki}^{TFR} & \text{if } Bad \leq q_i \leq TFR_i \\ \sigma_{ki}^{TFR} - \frac{(q_i - TFR_i)}{1 - TFR_i}(\sigma_{ki}^{TFR} - \sigma_{ki}^{Perfect}) & \text{if } TFR_i < q_i \leq Perfect \end{cases} \quad (6)$$

In this way the accuracy approaches a minimum when the quality level q_i is close to the TFR_i . This happens because the dispersion of the estimation $\hat{q_{ki}}$ given by σ_{ki} is maximized. In this case the estimation error $|q_i - \hat{q_{ki}}|$ is most likely to take larger values.

During simulation, after the inspection, the set of estimated quality values given by $\hat{Q} = \{\hat{q_1}, \hat{q_2}, ..., \hat{q_N}\}$ is attached to the data structure of every item. In the accepted or discarded units, the values of Q and $\hat{Q}$ are compared, and the statistical accuracy of the overall process is accumulated.

2.3 Numerical Simulation Software

What follows is a description of the System Simulation Object-Oriented Library tool called SOL. SOL was created to facilitate the task of implementing numerical simulations of QCP. SOL is a library of classes

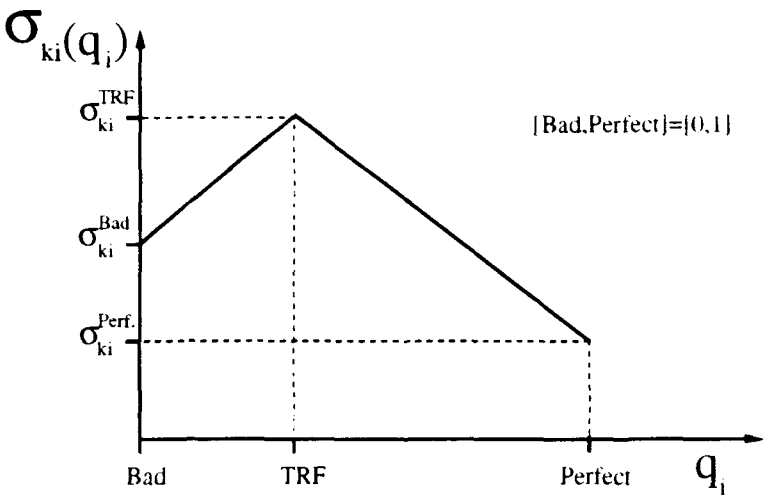


Figure 5: Standard deviation of estimated quality: Inspector k, defect i.

implemented in C++ [23, 60]. These classes contain models of the basic units as described previously. Special functions were added to the library which help to define the QCP and carry out the simulation.

When implementing a simulation, the classes are used to define objects that behave numerically as the basic units described earlier. Objects are closed structures that contain data and internal functions called methods. The Object's data are the variables required to model the internal state of a unit. Methods access and modify the internal data, hence, methods model the dynamic behavior of the basic unit. Some special functions are used to interconnect the objects in order to reassemble the connectivity of simulated processes. Other special functions are used to change default parameters or execute the simulation.

Rather than discussing the intrinsic details of SOL's classes, their use is described. An example is presented as an illustration. This simple case is also an introduction for the more robust application presented later.

2.3.1 The System Simulation Library Tool "SOL"

The implementation of a simulation using SOL involves the following steps:

1. Include the header files *sol.h, bclasses.h, sim.h*

2. Define all the objects representing the basic units that are required to assemble the plant. This is done by using the predefined classes in SOL.

3. Assign identification numbers to the objects and redefine the default characteristics. This is done by using the SOL function *install.* All units that are going to be used during simulation must be installed.

4. Define the connectivity of the plant. This is done by means of the SOL function *connect. Connect* establishes a direct unidirectional interaction connection between two units.

5. Configure basic parameters. For example: the number of defects with the function *defect(N)* and the TFR_i by assigning new values to the array TFR[].

6. Establish the initial, final and step simulation times as well as the progress report formats. This is done by means of the function: *simul_time*, and *prog_report.*

7. Execute the simulation using the function *simulate.* This function is also capable of performing step 6.

Many other functions furnished by the user can be added to SOL allowing step by step simulation, trap operation or other customized feature wanted.

To compile the final executable code the files *bclasses.c* and *sim.c* could be compiled (C++ compiler must be used) together with the simulator file. For instance, if the simulator is called *simul.c* the compilation command is:

c++ -o simul simul.c sim.c bclases.c

Otherwise, the simulator can be compile just by linking the source's object file with the library *sol* as follows:

c++ -o simul simul.c -lsol

Details on compiler commands can be found in the man-pages of Unix [2]. A simple but robust application of SOL is now given.

2.3.2 Example: Small Plant Simulation

The schematic for the plant is shown in Figure 6 This case is a simple quality control process that makes use of all the basic units defined before. The process is a single inspection line. After inspection, the accepted items go to the *buffer_A* unit then through the *transfer_A* unit and end in the accepted unit. Rejected items do the path starting at the *buffer_B* unit and ending in the *discarded* unit.

The C++ code for this workcell using SOL is as follows:

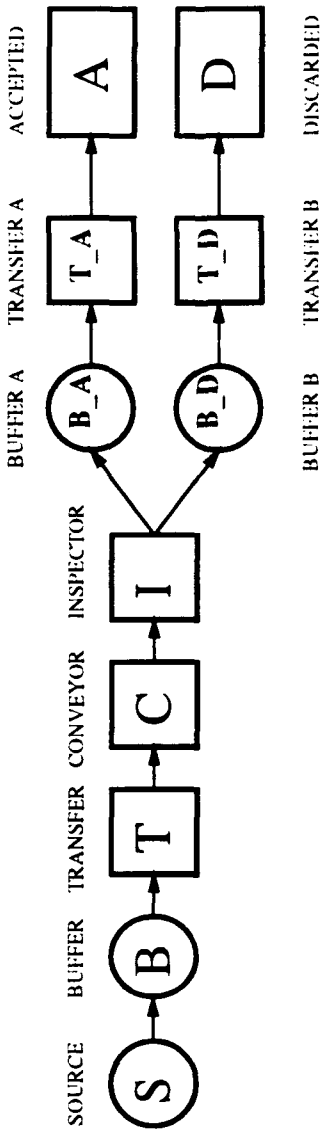


Figure 6: Single line workcell

```
// Default ************************************************
#include <stdio.h>
#include <stream.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>
// STEP 1 *************************************************
#include "sol.h"
#include "bclasses.h"
#include "sim.h"

main()
{
  // STEP 2 *************************************************
  // Define the units (Objects)  that are going to be used. //
  // change default parameters  (as listed here)
  Source       source(1);      //issue rate 1 item/sec
  Buffer       buffer(100);    //capacity 100 item
  Buffer       buffer_A(100); //capacity 100 item
  Buffer       buffer_B(20);  //capacity  20 item
  Transfer     transfer(1,1); //capacity 1 item, sped: 1sec/item
  Transfer     transf_A(1,3); //capacity 1 item, sped: 3sec/item
  Transfer     transf_B(1,3); //capacity 1 item, sped: 3sec/item
  Conveyor     conveyor(4);    //capacity 4 item
  Inspector    inspector(1);  //insp. time 1 sec.
  Accepted     accepted;       //None.
  Discarded    discarded ;     //None.

  // STEP 3 *************************************************
  // Install the unit and assign a ID number //
  install( &source,    10 );
  install( &buffer,    20 );
  install( &buffer_A,  21 );
  install( &buffer_B,  22 );
  install( &transfer,  30 );
  install( &transf_A,  31 );
  install( &transf_B,  32 );
  install( &conveyor,  40 );
  install( &inspector, 50 );
  install( &accepted,  60 );
  install( &discarded,  70 );

  // STEP 4 *************************************************
  // Connect the units //
  connect( 10, 20 );     // source -> buffer
  connect( 20, 30 );     // buffer -> transfer
  connect( 30, 40 );     // transfer -> conveyor
  connect( 40, 50 );     // conveyor -> inspector
  connect( 50, 22, GOOD );   // inspector -> buffer_A
  connect( 50, 21, BAD );    // inspector -> buffer_B
  connect( 21, 31 );     // buffer_A -> transfer_A
  connect( 22, 32 );     // buffer_B -> transfer_B
  connect( 31, 60 );     // transfer_A -> accepted
```

```
  connect( 32, 70 );     // transfer_B -> discarded

  // STEP 5 ************************************************
  defects(4);
  TFR[0] = 0.01 // 1% defective
  TFR[1] = 0.01 // 1% defective
  TFR[2] = 0.01 // 1% defective
  TFR[3] = 0.01 // 1% defective

  // STEP 6 and 7 ******************************************
  // Carry out the simulation from time 0.0s to time 30.0s
  // in steps of 1.0s.
  simulate( 28800.0, 1, FALSE );
}
```

The parameters of the function *simulate* specify the final time, simulation step and result report format. The time used in this code corresponds to one complete working day (8h shift). In this case the default parameters are changed during the instantiation of the units (definition of the objects). Note that when connecting the inspector, a third parameter has been used. This parameter determines the kind of items that can go through the connection. In this case *buffer_A* takes the accepted items while *buffer_B* takes discarded items.

The parameter FALSE in the function *simulate* indicates that the program should report results at the end of the simulation and not at every step.

The results report is presented in the default way as follows:

```
*** QCP Simulation using SOL V1.0 ***

-------------------------------------------
 Unit Type  |   ID   | Input | Output | Kept |
-------------------------------------------
Source          10       -       14500    -
Buffer          20       14500   14400    100
Buffer          21       7180    7180     0
Buffer          22       7220    7200     20
Conveyor        30       14400   14400    0
Conveyor        31       7180    7179     1
Conveyor        32       7200    7199     1
Inspector       40       14400   14400    0
Transfer        50       14400   14400    0
Accepted        60       13768    -       13768
Discarded       70       576      -       576
-------------------------------------------
```

```
------------------------------------------------------------
Inspector success rate (total inspected/successfully inspected)
  ID  |       rate        |
------------------------------
  50    14400/14397 (99.97%)
------------------------------
---------------
Source stall
  ID  | stalls |
---------------
  10    300
---------------
```

As shown in the previous example, implementation of QCP simulations is straightforward if SOL is used. The previous simulation was run in a SUN 4/380 workstation, and took 10.3 sec to complete. A simulation of an existing, (i.e., real) QCP is now given.

2.4 Application

The numerical simulation of an existing plant will now be described. The template for the initial simulation is taken from a factory located in Texas that gave considerable amount of production and quality control information to this project.

First, a brief discussion of the Texas plant is given. Then, there is a discussion of how SOL is used to create a simulated model of that template plant. The program details are followed by a discussion of the parameters chosen for the different units. The parameters are chosen so that the simulated model behaves numerically close to the real workcell.

At this point, the reliable simulated model is extrapolated so it matches average performance. Such an average is obtained by combining the production performance of other two existing companies. The numerical model obtained by the previous process is then extrapolated to the case of robots and machine vision-based inspection workcells. The parameters for these units are obtained from the experimental set up (see Section Five) and from commercially available robot performance tables. Lastly, there is a discussion of the simulation results.

2.4.1 Simulation of the Texas Plant

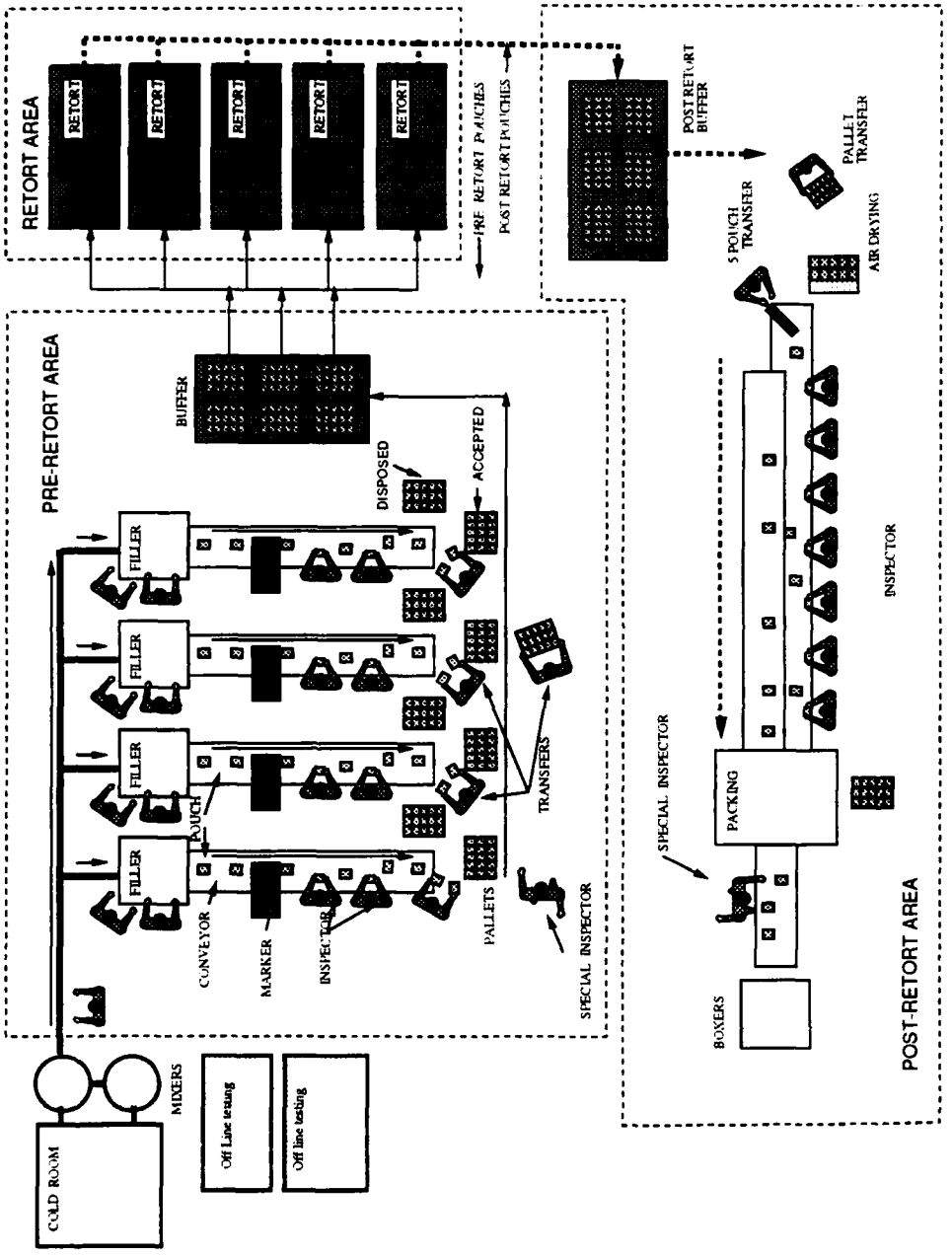


Figure 7: The Texas plant: overall organization.

The overall floor view of this plant is shown in Figure 7. Pouches (items) have to undertake two 100% in-line inspections. One inspection takes place before the retorting process and one after retorting. The two inspection zones are outlined in the figure. Two independent simulated models are created for the the two inspection processes. In this case the retort process is lengthy so it has large buffers before and after the process. This condition decouples it from the other two processes. For this reason independent simulations give the same results as the unified simulation. The simulations carried out for the two processes will now be given.

2.4.2 Pre-Retort Workcell Model

Figure 8 shows a single pre-retort inspection line and its corresponding simulation model. The simulation model is just a one to one correspondent of the real inspection line using the basic units described in 2.2. The filler is replaced by a source unit emitting pouches at the same rate. Pouches are transported by the conveyor belt up to the two inspectors. After inspection, inspectors return all pouches to the conveyor belt; however, in the real plant, discarded pouches are put on the conveyor with a 90° rotation with respect to the original orientation. In the simulation model pouches have a variable where inspector units write the quality control test results. It is up to the transfer operators (real case) or units (simulated model) to classify them into two post-inspection buffers.

The complete pre-retort inspection line is shown in Figure 9. It has four inspection workcell in parallel with a total of 42 units distributed as follows:

- 4 *sources.*
- 8 *inspectors*, two per line
- 4 *conveyers* before the couple of *inspectors*, and four *conveyers* after.
- 4 *transfers.* Moving and selecting pouches after inspection.
- 8 *buffers.* Two per line, one for accepted and one for rejected items.

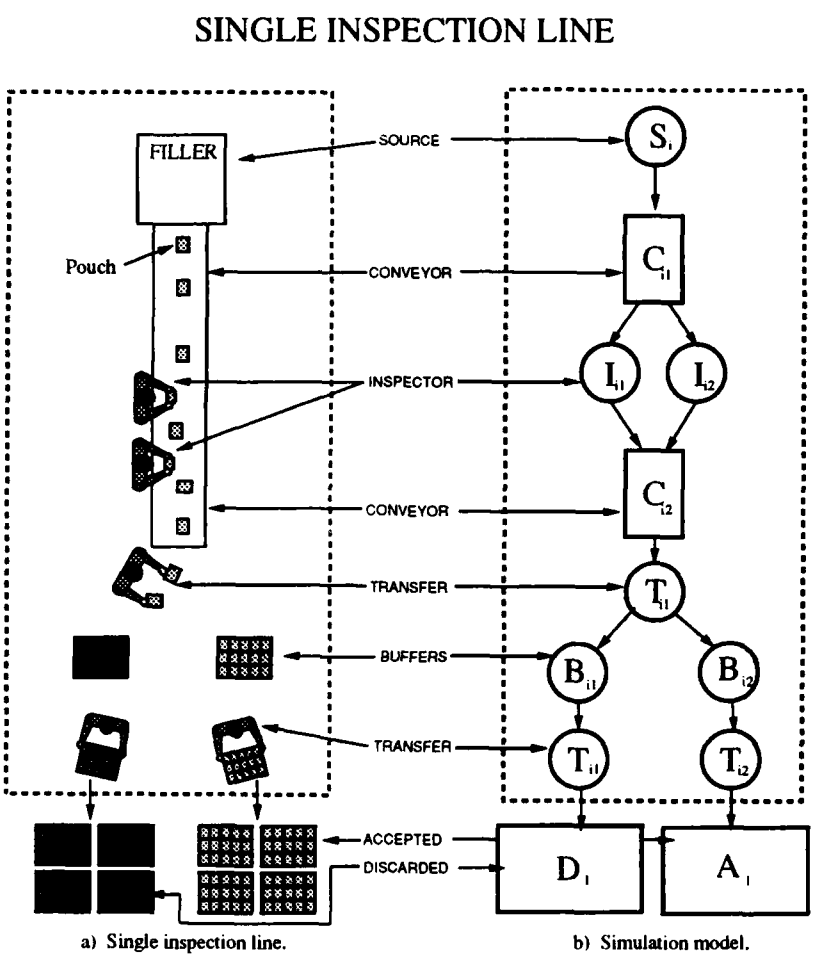


Figure 8: Pre-retort inspection: Single line model.

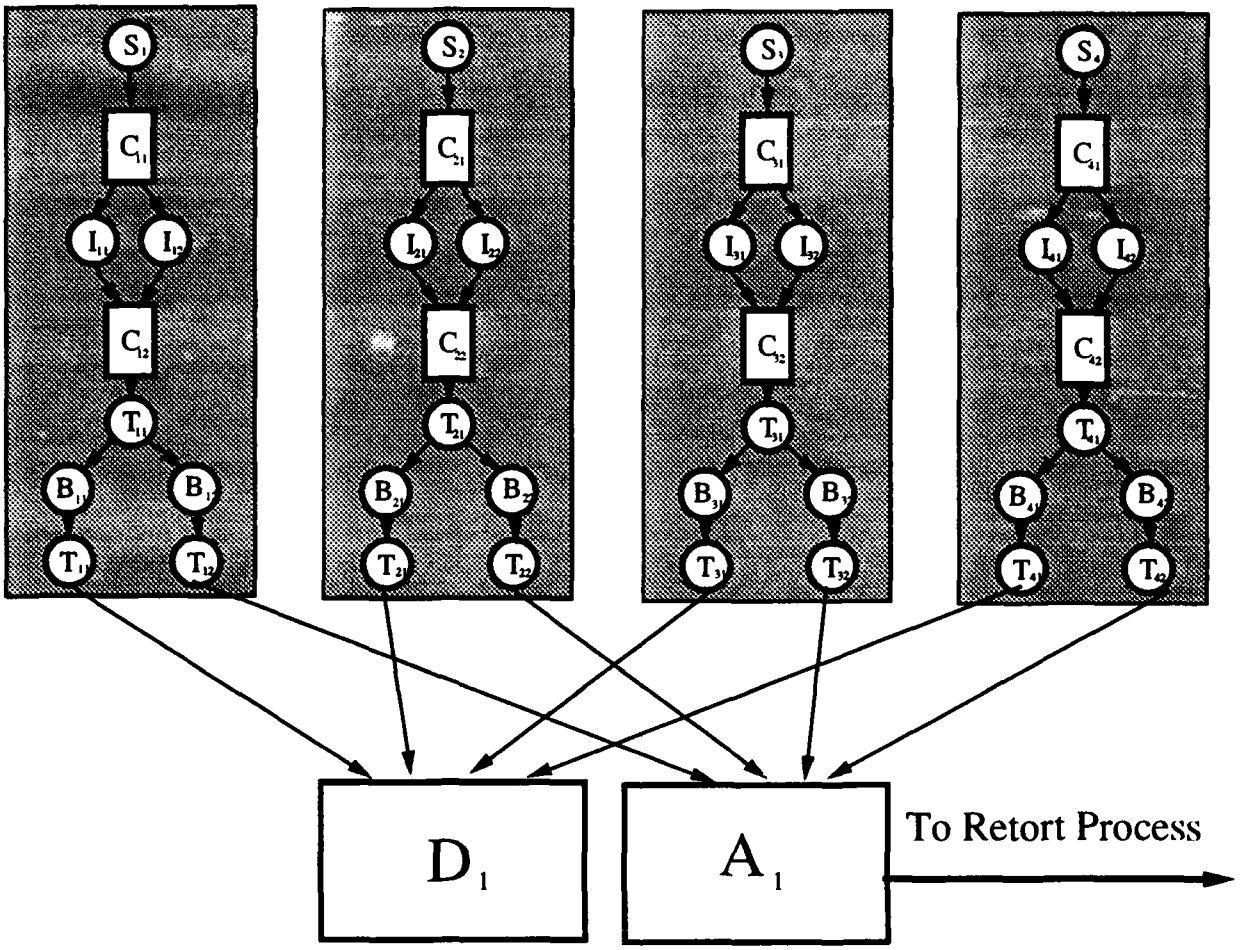


Figure 9: The Texas plant: Complete pre-retort inspection lines.

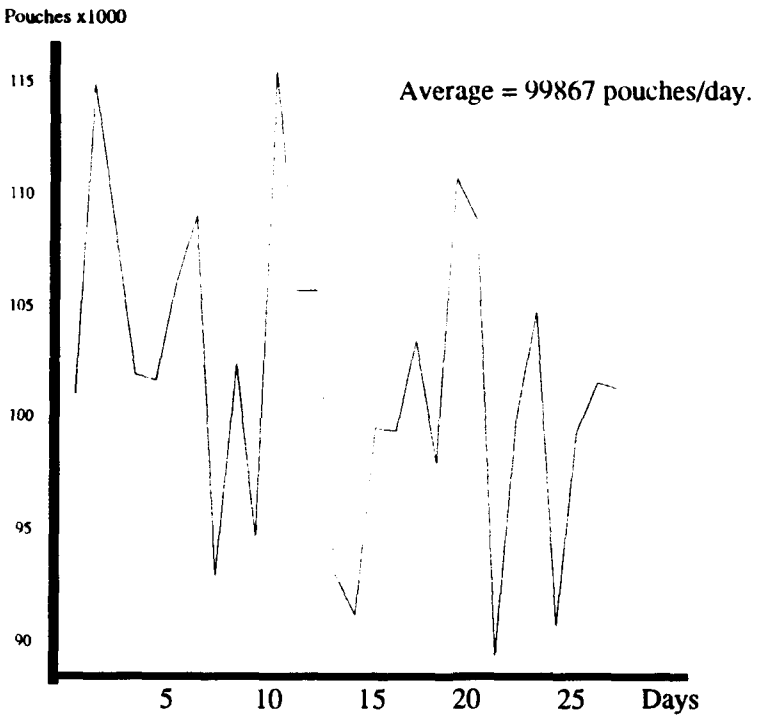


Figure 10: Texas plant: Total daily production.

- 8 *transfers*. Two per line moving the pouches to their final destination, the *accepted* and the *disposed* units.
- 1 *discarded* unit.
- 1 *accepted* unit. This one replaces the real retort process and the real buffer before that process.

As it was shown in the previous section, the implementation of a simulator for this plant is straightforward if the SOL is used. The implementation is just a matter of defining the plant components and describing their connectivity by mean of SOL's objects (*item, source, etc*) and functions (*install, connect, etc.*) However, to match the performance, the parameters of the simulated model have to be fine tuned. The code for this simulation is presented in Appendix A.

Figure 10 shows the performance of the template plant for a 28 days production run. The curve represents the total number of pouches being inspected each day. The average daily production is 100,000

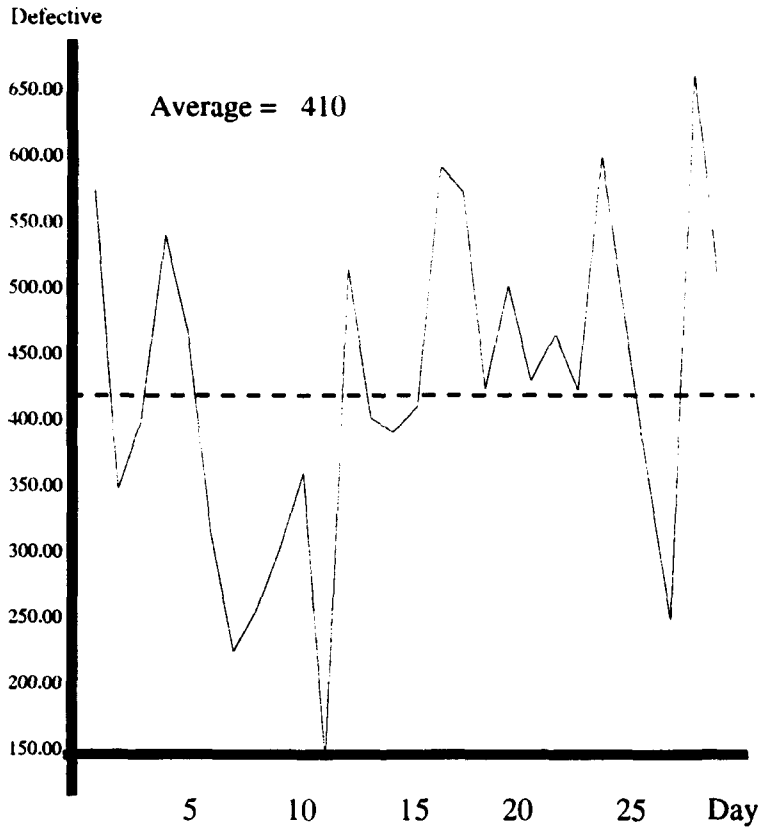


Figure 11: Texas plant: Defective pouches in pre-retort.

pouches. Figure 11 shows the total number of defective pouches in the same interval of time. The average number of discarded pouches is 500, so the percentage of defective pouches is then around 0.5%. In all the data gathered, no specification was made about the independent type of defects. That is why the numerical simulation implemented here uses only one defect with $TFR = 0.005$.

In order to obtain similar results with the simulation model the following parameters are chosen:

- Source: Rate =1.5 pouches/sec
- Inspector: Rate=0.347 pouches/sec, inspection time 2.9 sec.
- Conveyor: Capacity 10 pouches, transport time 10 sec.
- Buffer: Capacity 100 pouches for the one before the accepted unit, and 15 pouches for the one before

the discarded unit.

- Transfer: Capacity 1 pouch, transport time 1.5 pouches/sec.

It is important here that the Texas plant is one of the most efficient in the country. The following is the performance of other two companies:

- Company II: Four pre-retort stations, about 80,000 pouches per day. 10% defect rate.
- Company III: 52,000 pouches per day. (defect rate unknown)

Unfortunately, detailed information about the quality control components and the workcell connectivity of these companies was not available. Hence, the general layout of the Texas plant will be used together with the **average** performance of all three companies in order to have a representative *generic simulation model.* The average to be matched is then around 70,000 pouches per day with a discarded rate of 5%. This means that out of the 70,000 pouches 3,500 are rejected.

The simulation of the pre-retort workcell matches these output results if the average inspection time is set to 3.9 seconds (for all the inspectors) assuming a shift of 8 hours a day or 4.2 seconds for a 10 hour shift. Nevertheless, this inspection time is a function of the topology (components plus connectivity) of the plant. As it was mentioned, the Texas plants was used as a prototype for these matters.

The *TFR* was set to 0.05 and only one defect was used. This condition makes sure that 5% of the pouches are defective. Each one of the four source units was set up to generate the same amount of pouches a day (1/4). This condition balances the load in each line an makes the analysis easier. However, for more sophisticated analysis, it is possible to break this symmetry condition and study carefully its implications.

2.4.3 Automatic Pre-Retort Workcell

STP11 deals mainly with the feasibility of automating the inspection process. That is: sources conveyers and some transfer units may change a little in the automatic version. However, inspectors will be replaced

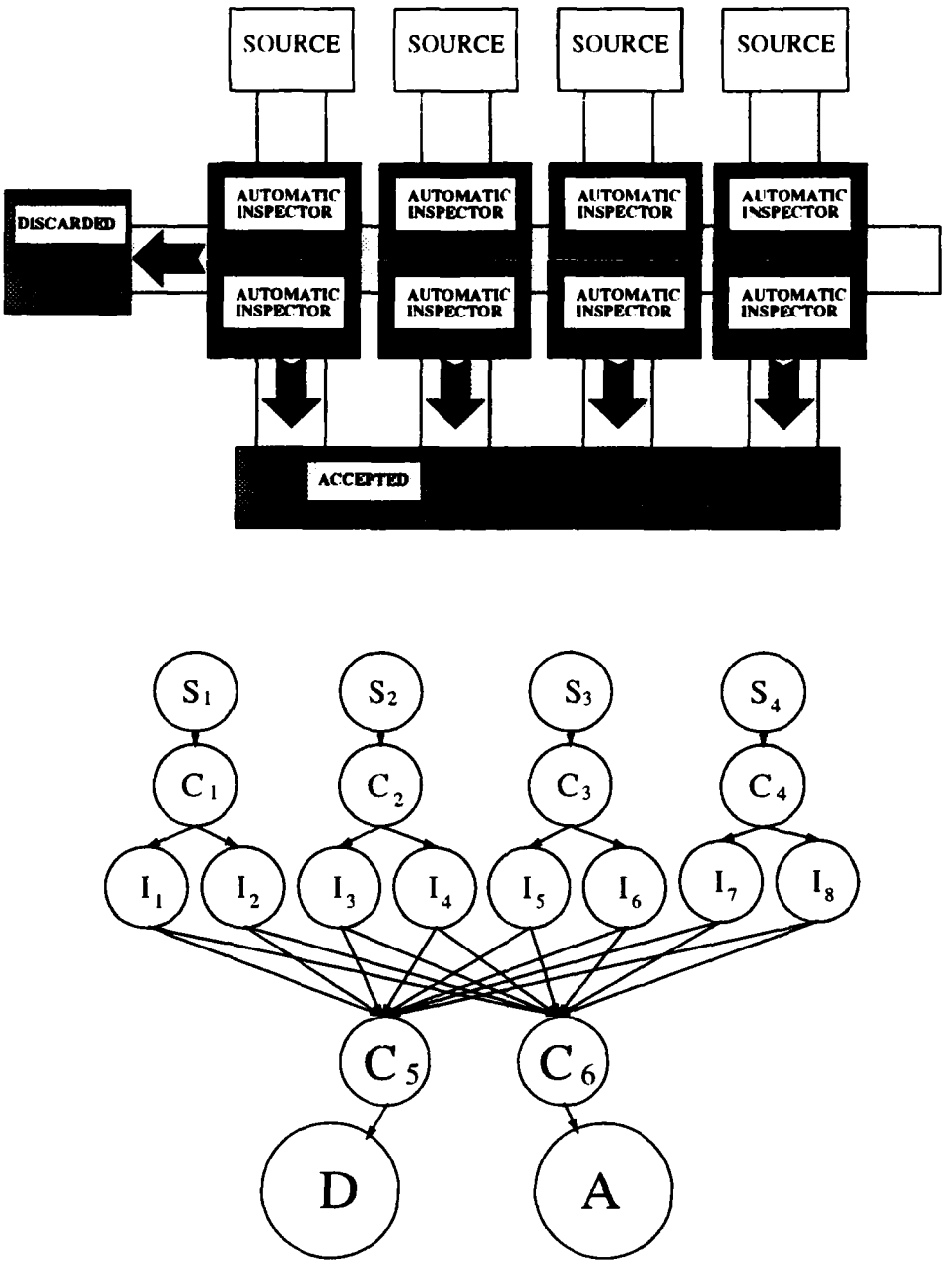


Figure 12: Automatic workcell for pre-retort.

completely. For this reason the main parameter for performance comparison is the inspection time and the inspection accuracy.

Inspection accuracy can be assumed to be greater or equal to that of the human inspector. That is because as it will be shown, the inspection time can be compensated by increasing the number of inspectors working in parallel. However, the accuracy can not be increased in a simple way. For instance, the pouch can be double checked, but this will double the implementation's cost and complexity. For simulation purposes it is assumed that the accuracy is close to perfect and the attention is focused on the inspection time (critical for the plant's throughput).

It will be shown in Section 5 how the inspection time of the machine vision system is larger than the human inspection time. This is because a reliable system requires a time consuming image processing. Although the inspection time for the machine vision obtained in preliminary studies (see Section 5) is still relatively high (14 seconds), the inspection time used for the workcell simulated here is assumed to be 8 sec. This is because (as it will be discussed in the conclusion of Section 5) there are many other techniques (not studied presently) that will, in the future, lower substantially the inspection time.

Figure 12 shows an alternative pre-retort workcell based on automatic inspectors. The workcell layout is quite different from the human-based workcell but the parameters for sources, conveyors and buffers remain basically the same. The transfer units are omitted because it is assumed that robotic inspectors with several degrees of freedom are capable of performing complex transfer movements. The intermediate buffers are omitted because computer controlled units can optimize the synchronization of task execution. Another difference is that the inspection process is carried out for 16 hour per day. With these characteristics, this alternative automatic workcell has the same performance as the *generic simulation model.*

2.4.4 Post-Retort Workcell Model

Figure 13 shows the inspection line and the respective simulation model for the complete post-retort inspection. The units involved in this model are:

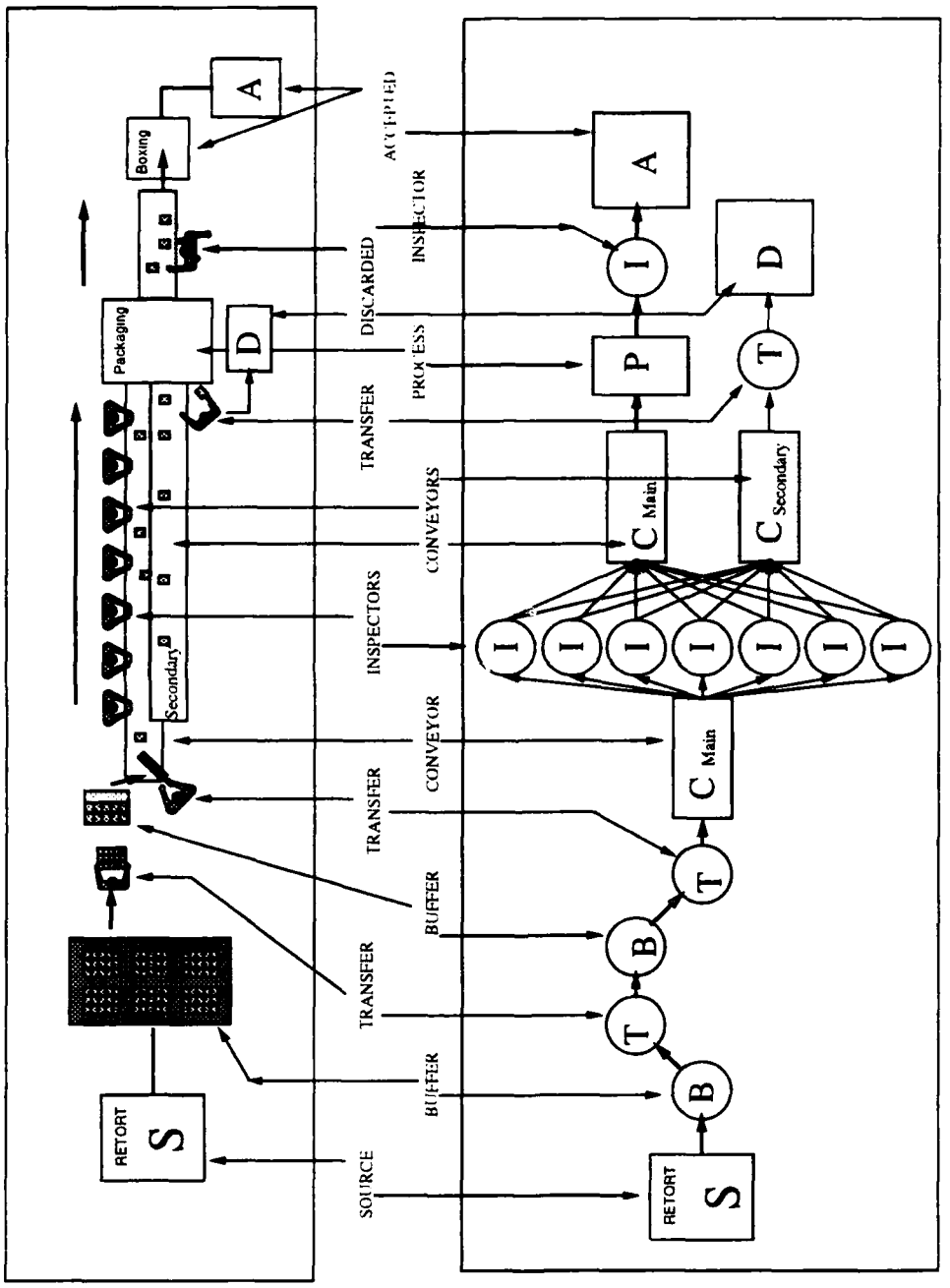


Figure 13: Post-retort workcell and its model

- 1 *source*, replacing the retorting process.
- 1 post retort *buffer*.
- 1 *transfer* between the large post-retort *buffer* and a small pre-inspection buffer. (15 pouches).
- 1 pre inspection *buffer* mentioned above.
- 1 *transfer* unit capable of moving 5 pouches at the same time.
- 1 *conveyor* to transport the pouches to the inspector.
- 7 *inspectors*. Inspecting pouches from the main *conveyor*. Accepted pouches are put back on the main *conveyor* while discarded pouches go to a secondary *conveyor*.
- 2 *conveyors*. A main one for accepted pouches and a secondary one for discarded pouches.
- 1 *process* (packaging). Processing accepted pouches from the main conveyor.
- 1 *transfer* unit that takes discarded pouches from the secondary conveyor to their final destination, the discarded unit.
- 1 final *inspector*.
- 1 *accepted* unit. This one includes the boxing process that has nothing to do with the overall inspection process.

The post-retort processes may look quite different than the previously analyzed pre-retort processes. Nevertheless, from the simulation point of view, these two processes are quite equivalent. In fact, the total number of inspectors is almost the same as the Texas plant. The main difference are the source units. Pre-retort processes require several fillers working in parallel to produce the right amount of pouches. The post-retort source is just a large buffer where the pouches coming out of retorting are being stacked.

It was previously mentioned that the critical components of the QCP are the inspector units. The post-retort case may require a different defect detection approach. This is because new types of defects could appear after the heating process. For the simulation presented here, this matter was not explored because detailed information was not available from the industry.

The automatic workcell model proposed for post-retort is the same as the one for pre-retort. However, the source units for this case can be assumed to be in charge of the air-drying instead of the pouch filling and sealing (see Figure 7). It should not make a difference as far as the output rate is conserved.

The code for the post-retorting simulation is listed in Appendix A for future reference.

2.5 Conclusions

The quality control process was decomposed into several simple elements. Numerical models for each elements were easily elaborated based on empirical knowledge and common sense. The abstract models were implemented in the object oriented language C++ generating a library of classes (objects). This partitioning of models into objects represented a powerful tool for numerical simulation. Simulation for simple and complex workcells were implemented with few lines of code.

A simulator was built to resemble both numerically and topologically an existing plant from which a large amount of information was available. The simulator was extrapolated to match the average performance of other two plants. A fictitious (non-existing) robotized plant was proposed to perform as well as the average template plant. Hence it was proved that there are enough controllable parameters to design a robot-based plant that performs as well or better that a human-based plant. However, it is important to observe that the robot based inspector must at least match the error-rate of a human operator.

Even though sufficient data was not available to create a detailed simulation, it was shown that numerical simulation is a powerful tool for future design. The numerical model can be extrapolated to extreme conditions and the effects on individual components can be analyzed. Such flexibility is in many cases absent in real prototypes.

3 Graphics Simulation

3.1 Introduction

A graphics simulation was created to complement the numerical simulation, called the Interactive 3D Graphical Simulation IGAS. The simulator processes are distributed among two computer workstations. The graphics rendering is implemented on a HP9000/360 graphics workstation with graphics accelerator, while the numerical simulation and the user interface was implemented on a SUN 4/360 workstation. The simulation IGAS was developed in the Human-Machine Interface Laboratory where several tools for virtual reality simulation are available.

IGAS displays animated models of a "human-based" and "robot-based" inspection workcell. The user's interaction with the 3D graphical models is performed by means of a track-ball device (Dimension6 Ball [15]), which allows the user to virtually "fly-through" the 3D graphical models.

To increase the simulation performance, the numerical simulation and the Dimension 6 Ball drivers run on a SUN/4 workstation while a server-client scheme transfers the required data to the Hewlett Packard workstation via an ethernet network.

3.2 Graphics Simulation

3.2.1 General Description

Human-Based Workcell The human based station is shown in Figure 14. The block diagram of such a station is shown in Figure 15. It consists of two inspection stations, one before and one after a generic process. The pouches are generated in a source unit and transported by the first conveyor. Four inspectors, two at each side of the first conveyor, visually inspect the pouches. The pouches accepted by the inspectors are placed back on the first conveyor.

The rejected pouches, at this first inspection area, are placed on a second conveyor which is perpendicular to the first one and goes underneath it. This conveyor takes all the defective pouches to the *discarded* unit (not shown in the graphics simulation).

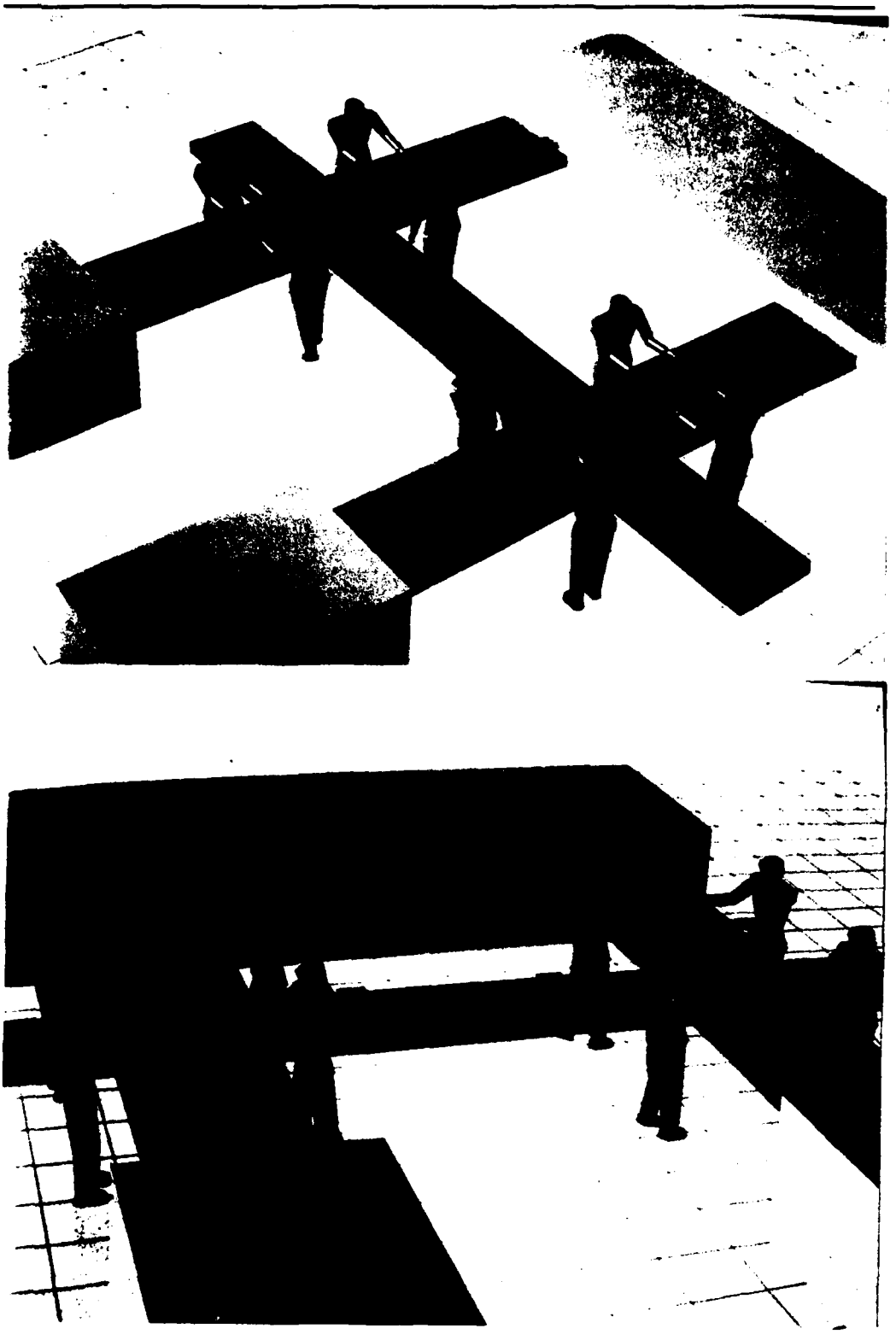


Figure 14: Human based workcell: Screen picture

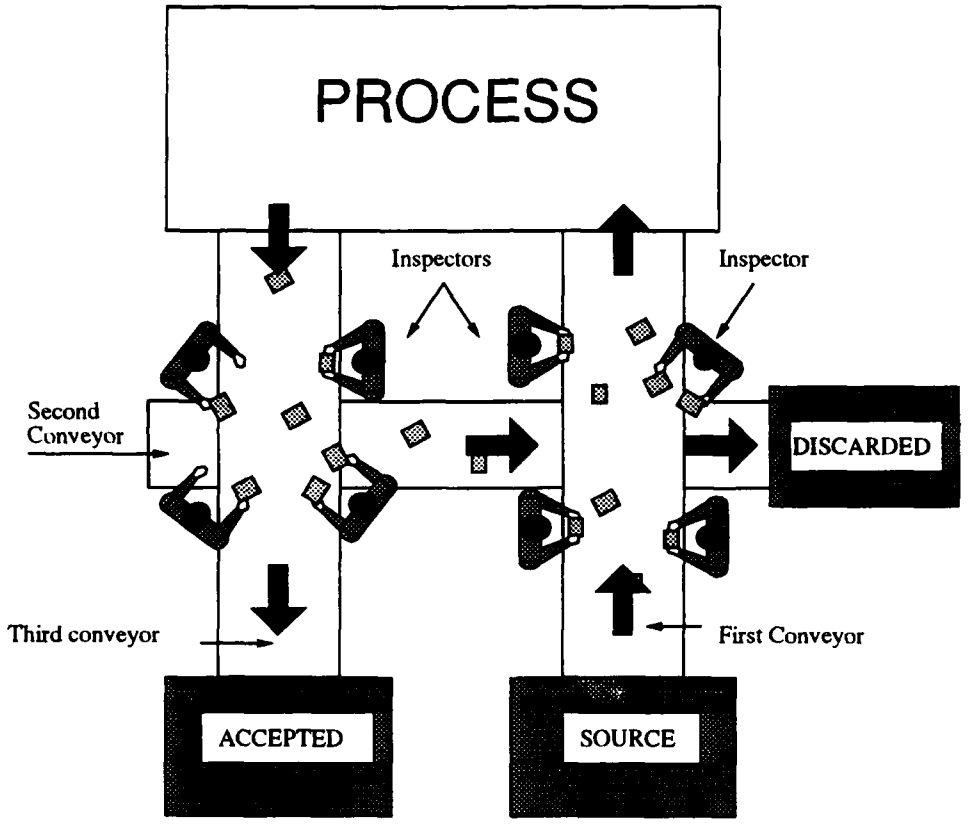


Figure 15: Human based workcell: Block diagram

A third conveyor transports the pouches from the generic process to the second inspection station. This inspection area has also four inspectors and shares the same conveyor for discarded pouches. The accepted pouches will remain on the third conveyor which transports them to an *accepted* unit.

Robot-Based Workcell The robotic workcell can be considered as a flexible-automation implementation shown in Figure 16. The disposition of the overall workcell consists of two parallel process lines. Each line having one inspection workcell before and one after a generic process.

Two robot manipulators are used to handle the pouches. In this particular model they work coupled to turn over the pouch. Such a process is required in order to allow the vision system to inspect both sides of the pouch. The vision systems, which inspect a single side of the pouch, are located before and after the robot arms.

The conveyor of each inspection workcell is synchronized with the inspection process. This is done so that the pouch remains steady (not moving) during the image acquisition process. To do this, the conveyor stops for some period of time. The robot manipulators are also synchronized so that when the conveyor stops there is also a pouch present underneath the second inspection camera system.

3.3 System Configuration

The overall system configuration for the graphics simulation is shown in Figure 17. Two dedicated workstations are used, a SUN 4/360 workstation and an HP 9000/360 SRX Graphics workstation with a 98766 Starbase graphic accelerator. The HP workstation creates the 3D graphics display, while the SUN workstation runs the drivers, the Dimension 6 ball controller and the network communication.

Processes on the same workstation communicate via shared memory segments while processes on different workstations communicate over the ethernet. For the latter, a server-client network scheme is employed. On one end of the ethernet, the SUN_server (running on the SUN workstation) reads the serial port to transduce the user commands on the Dimension6 Track-ball, then, it writes the appropriate data in the network.

Figure 16: Robot based workcell: Screen picture

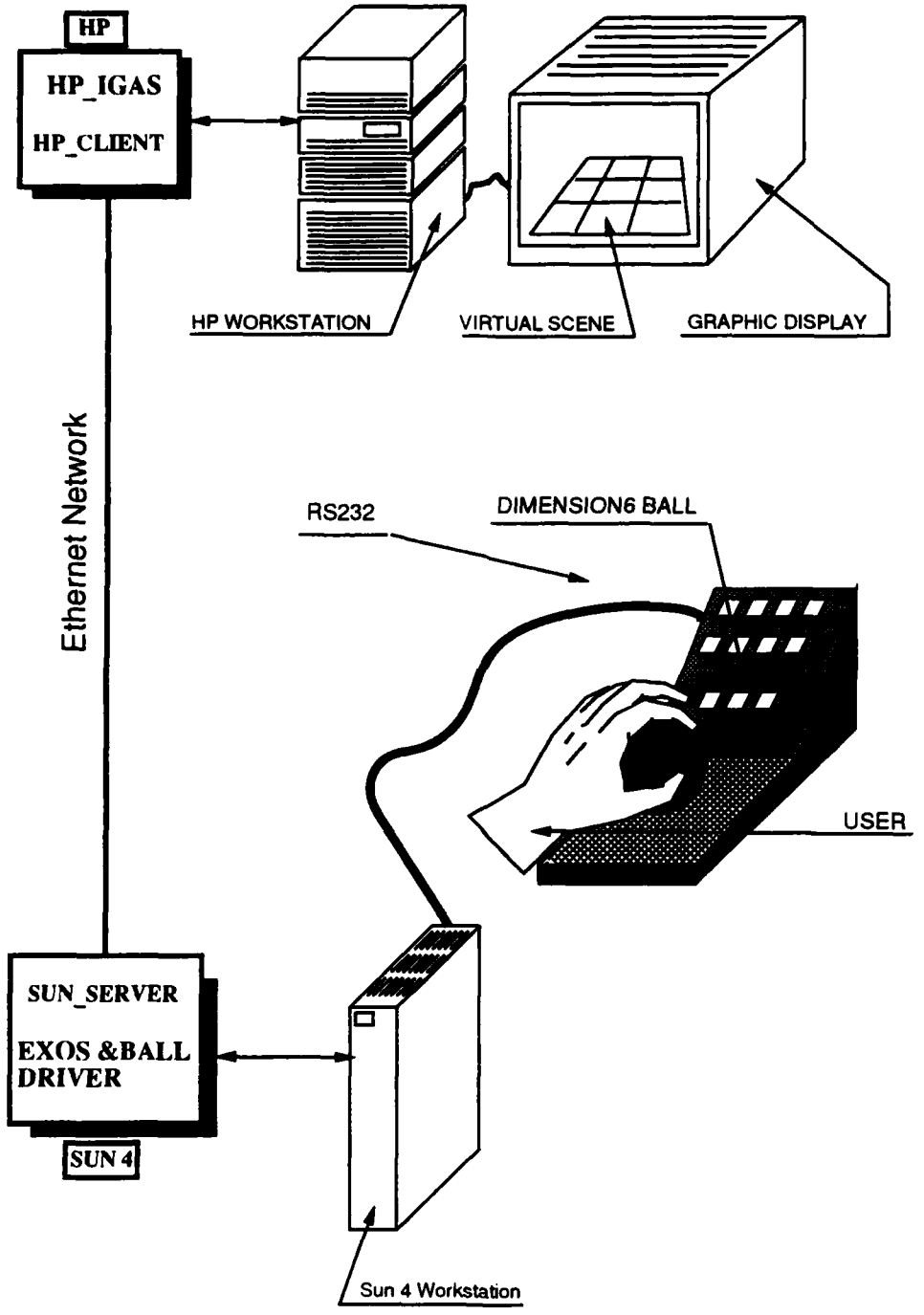


Figure 17: System configuration for graphical simulation.

The Dimension6 Track-ball controller uses optic sensors to measure forces and torques applied to the ball [15]. By gently squeezing the ball the operator applies forces and torques that are sent over an RS-232 serial link to the SUN. The Dimension 6 driver uses time rate control to map force and torque readings into positions and orientations of 3-D objects.

To compensate for different control needs, the track-ball keypad has been used to change the force/torque gains G. In this way by using a high gain the user can fly faster through the simulation. A small gain G is used when precision positioning (low speed) is required. Six integers (of one byte each) are read from the track-ball optic sensors. The values correspond to three forces F_x, F_y, F_z and three torques M_x, M_y, M_z. These values are multiplied by the gain factor G and then sent through the network to the graphics application. G changes from 0.2 to 1.0 in steps of 0.2 as selected by the user using the track-ball's keypad.

At the other end of the network, the HP_Client (running on the Hewlett Packard graphics workstation) reads the ethernet and stores the received data in the corresponding shared memory segment. This information is then used by the graphics routines HP_IGAS to steer the view-point (camera point of view) and simulate a real-time flight-through in the graphics model. In the graphics code, the forces GF_x, GF_y, GF_z are used to displace the viewer along the viewer coordinate system. The torques GM_x, GM_y, GM_z are used to rotate the camera. An alternative simulation mode uses the same data not to move the camera (which remains fixed) but to move the graphics model.

3.4 Graphics Simulation Code

To improve the graphics refresh rate, a Display List technique is used together with the HP Starbase graphics library [34]. A special data structure called "segment-network", was created to store the model of the virtual scene. Polygonal objects and their corresponding 4x4 homogeneous transformations [68, 34] are placed in the segment-network in a hierarchical way. Labels are placed in the segment-network at the position of transformations of the objects that are going to be animated.

Animation consists of editing the segment-network, traversing the segment-network to update the transformed object and finally displaying the resultant image. Editing the segment-network is performed

by changing transformation matrices referenced by their labels. In this way objects can be rotated and translated dynamically.

During animation, once the segment-network has been edited, the history of the whole data structure is refreshed traversing the hierarchically arranged segment-network from the higher-level to the lower-level. Transformations on a higher level are applied to all the other objects in the lower levels which belong to the same branch. In this way it is possible to move (translate or rotate) a complete set of correlated objects changing their parent (root) transformation [68]. The resulting image is stored in a secondary image buffer. Subsequently, the main buffer, holding the current image on the screen, and the secondary buffer are swapped. This results in the new image being displayed.

A dedicated data base was created which contains modeling information of the generic-objects: color, vertex list, normal to the surface and normal to the vertex. At the beginning of the simulation, the data base information is used to generate the objects in the segment-network. The same generic objects can be used several times to build similar parts in the segment-network. Such a building-block technique saves storing space and eases the global off-line changing of objects.

Scaling transformations are used to reshape the generic-object to its final form, a rotational transformation reorients child objects with respect to their parent object, and finally, a translation is in charge of placing the elements in their positions. Figure 18 shows a block diagram of the graphics simulation program.

The average refresh rate for the graphics simulation was 4 *frames/second.* If the viewer is relatively close to any object in the virtual scene, that object may require a large area of the image frame. In that case lower bandwidths of 1.2 *frames/second* are obtained. When the viewer is placed far away from the graphics model, so that the model uses a fraction of the screen, the graphics refresh rate reaches it maximum of 10 *frames/second.*

On the other hand the refresh rate is also a function of the complexity of the segment-network being displayed. For the case of the human-based workcell around 600 polygons were used while for the robot-

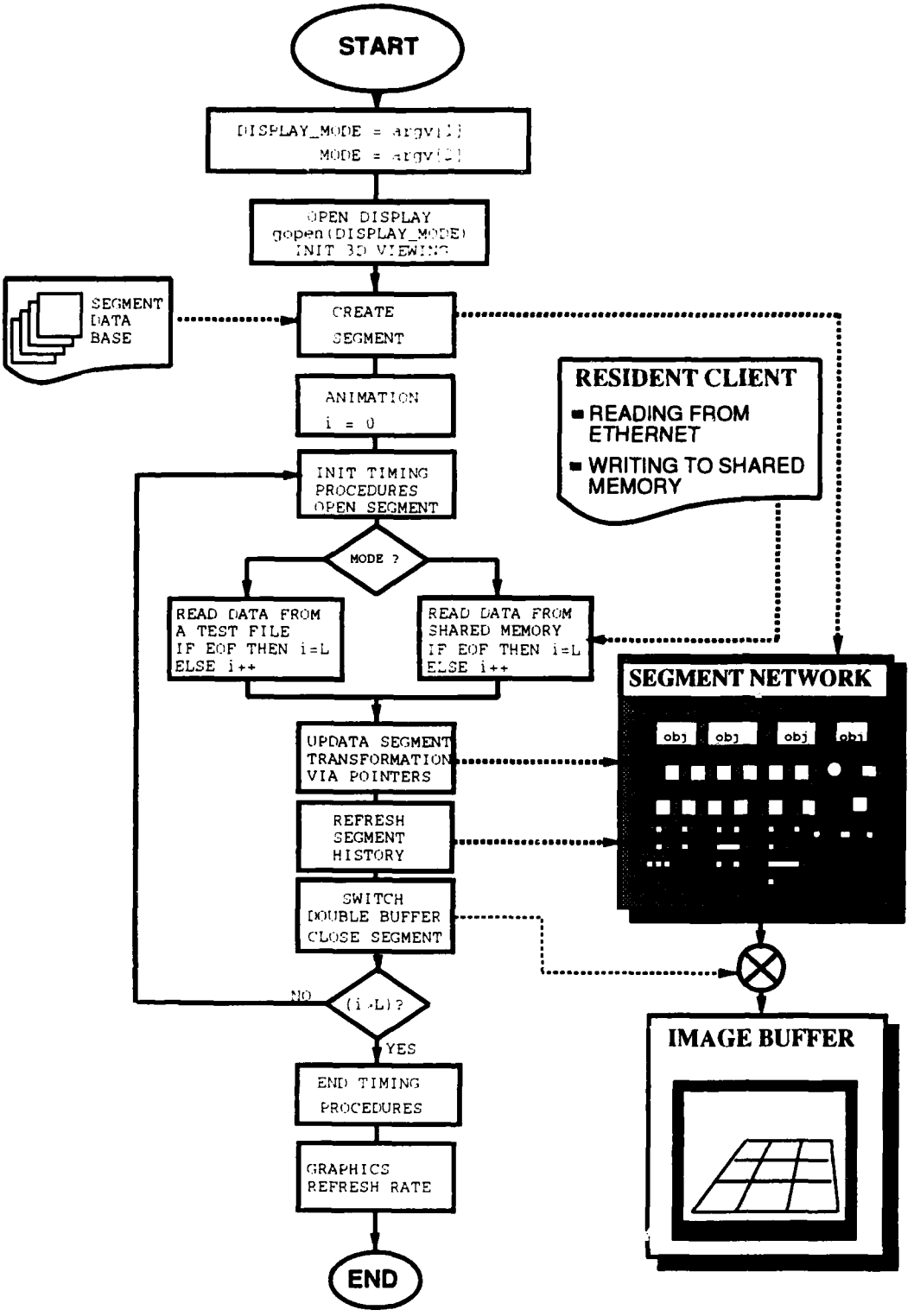


Figure 18: Graphical simulation program: Block diagram

based workcell only 300 polygons were required. For this reason the refresh rate for the latter was 30% faster. The low bandwidth obtained in this simulation was a consequence of the graphics workstation performance. For instance, if the same simulation is run on a HP 755 workstation it is possible to obtain 50 or more *frames/second.* Such a high bandwidth allows to implement more complex and realistic 3D simulation.

3.5 Conclusions

Graphics simulation is also a powerful tool for complex system feasibility studies. In this Section the qualitative aspect of the workcell's layout were easily visualized with the help of a graphics simulation (virtual plant). Interaction with a 3D model was done by means of a six degrees of freedom track-ball. Such control permitted the exploration of the 3D graphical model by simulating a user "flight-through".

The implemented graphics simulation allowed to compute the workspace requirements for human-based and robot-based workstations. For the examples studied in this simulation, the space requirements for robot workcell system (robot manipulator plus machine vision systems) are larger than those for human-based workstation. In the robot-based layout the pouch must be turned around before it undertakes the next vision inspection (in order to inspect both sides). This procedure is implemented by two coupled robot arms manipulators. Considering the pay-load requirements (100g-500g) such a device can be implemented in a small space. However the two vision systems may require a larger volume which can exceed a human operator's work space volume.

The main components of the robot-based workcell were graphically modeled in this Section. The robot manipulator and the machine vision system. Although the graphics simulation proved to be a very efficient tool, for a complete layout analysis it is required to model a specific design of the robotic workcell. To do that, the following two sections will analyze the two elements (robot-manipulator and machine-vision system) in detail. This will explain why the graphical model used the particular 3D design presented in this section.

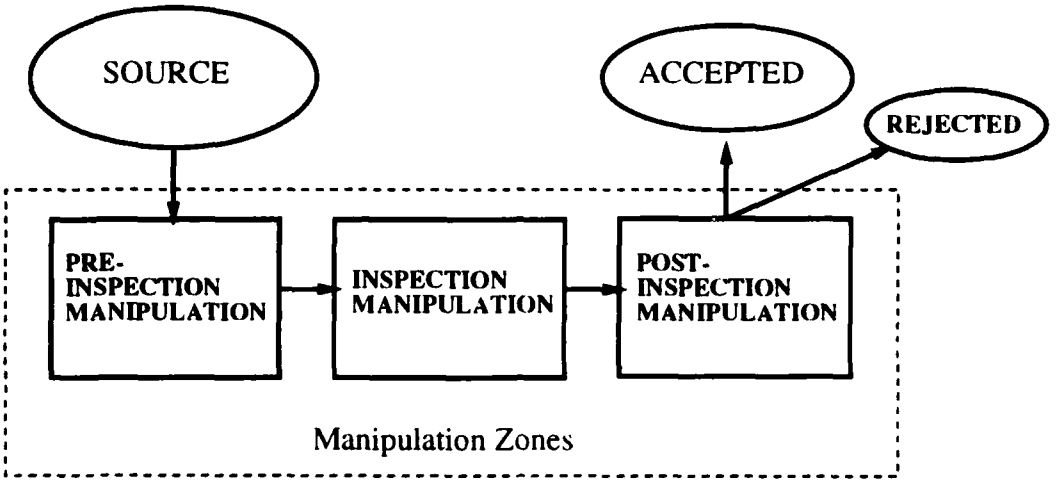


Figure 19: Pouch manipulation zones

4 Workcell Robot Manipulators

4.1 Introduction

This section presents the description of two robotic workcells which can be used for pouch manipulation. The vision system is described in the next section. However, the pouch manipulation needed for that system will be introduced here.

The workcell area is decomposed into several manipulation zones. The particular pouch handling required in each zone is discussed. Based on the overall pouch movement requirements and the manipulation zones, two robot models are proposed. For each case, a brief analysis is made in terms of the manipulator mechanism, the control systems required and their performances.

4.2 Pouch Manipulation Zones

Three main zones for pouch manipulation can be distinguished: the *pre-inspection*, the *inspection* and the *post-inspection* zones. In the pre-inspection zone the pouches are brought from sources or transport systems to the inspection zone. In the inspection zone the pouch is manipulated so that the vision system can take all the required images. In the post-inspection zone the pouches are classified as accepted or rejected and

returned to the transport systems or the buffers. Figure 19 shows a block diagram of the three zones.

What follows is a description of the relevant details of each of the pouch manipulation zones. Associated to each of the zones is a machine (a robot or a fixed-automation device) which handles the pouches. Such a machine will be called a *manipulator* and will be referenced together with its zone name, i.e.: *pre-inspection manipulator*, etc.

4.2.1 The Pre-Inspection Zone

In many cases, pouches are brought to the inspection zone via conveyors. In those cases the pouches usually come in random orientations and positions (along the conveyor). For human operators, locating and grasping a pouch from a conveyor requires neither training nor special abilities. In contrast, the same task could represent a real challenge for simple robot based workcells. For this reason, some of the automation models proposed in this research require that the incoming pouch be located and oriented between certain ranges. Otherwise, extra vision systems or sensing systems will be required so that the robot manipulators be able to perform precise grasping. Details on this matter are addressed when the particular models are described. However, it is important to point out that if any extra devices are required in order to locate or orient the pouches, such devices must be considered part of the pre-inspection manipulator.

4.2.2 The Inspection Zone

Human inspectors usually rotate or stretch the pouches during inspection to visualize hidden areas or to remove wrinkles. However, it has been established that the Vision System (developed by the STP11) requires only that the pouch be stretched out during the image registration process. The pouch must also be fixed and held steady during this process. Both sides of the pouch have to be available to the cameras for a two-side inspection process.

The two side problem can be solved in three ways:

- The inspection zone manipulation is designed so that a double Vision System can take images of

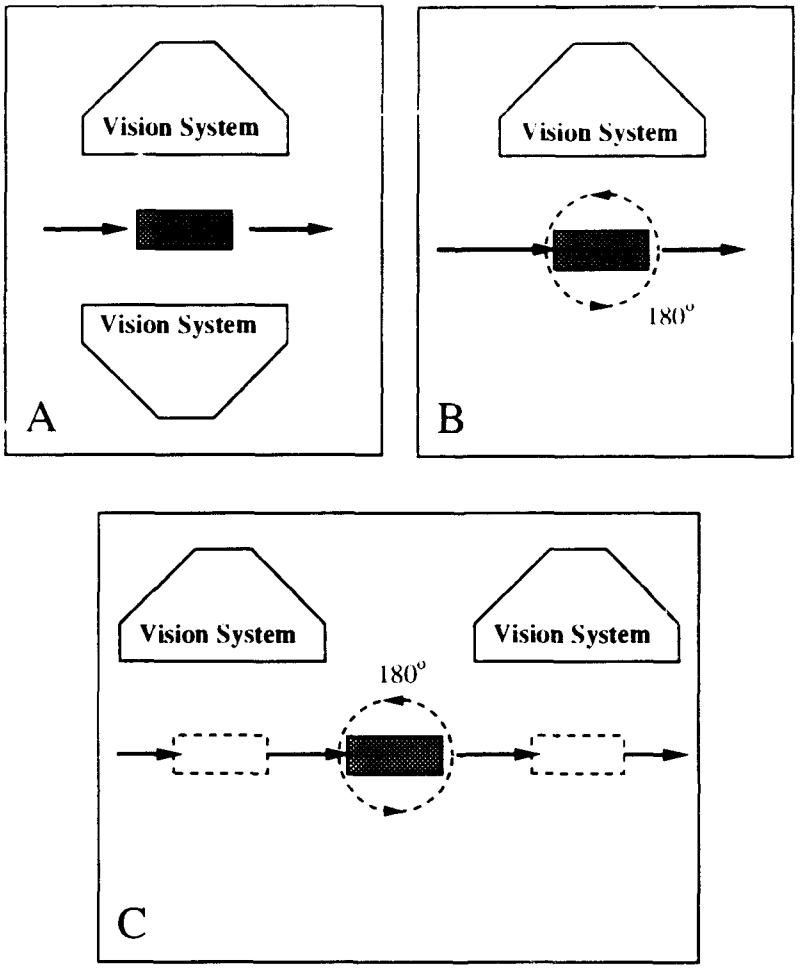


Figure 20: Pouch manipulation in the inspection zone

both sides at the same time.

- The inspection zone manipulation is such that the pouch is rotated 180 degrees allowing a single Vision System to "see" both sides of the pouch.

- The inspection zone manipulation has two consecutive Vision Systems and the pouches are turned over before they go through the second Vision System.

Figure 20 shows schematically the three possible manipulation methods for the inspection zone. For future reference, they will be called *inspection zone manipulation* types A, B and C.

4.2.3 The Post-Inspection Zone

This part of the pouch manipulation does not represent a major challenge. It consists basically of removing the pouches from the inspection zone and placing them into conveyors or containers. Two possible destinations have to be allowed in this zone. The accepted and the rejected pouches should take different paths. However, no special time should be spent on how the pouches are stored or stacked after this process. The main objective of the post-inspection manipulator is then to free the inspection manipulator from already inspected pouches.

Some manipulation models do not allow a clear distinction between zones. This happens when the same manipulator is used to perform pouch handling for two or three of the zones described above. For example: a robot arm used to pick incoming pouches from the conveyor, locate the pouches under the Vision System performing the stretching and rotation required for inspection, and then dropping the pouch back on the outgoing conveyor.

4.3 Types of Automation

The different levels of automation affect drastically the implementation. When automating a manual process there are two types of approaches that need to be considered: flexible-automation and fixed-automation.

To use a single multiple-degree of freedom robot arm to handle the pouch in all the manipulation zones is a typical case of a high-level flexible-automated workcell. In this case the manipulation zones represent just parts of the robot work envelope and not separate pieces of hardware. It is considered to be a high-level automation because the manipulator is a multi-degree of freedom arm with complex task and trajectory planning capabilities (given by its mechanical design and its control interface). It is considered a flexible-automation because most of the high-level systems are very flexible and programmable in terms of tasks and changing environments. The same robot can be adapted for different pouch sizes or different inspections requirements. It can be used in other places in the plant to do packing or palletizing.

In contrast to high-level flexible-automation is fixed-automation. In this case specialized hardware devices are devoted 100% to perform a particular part of the whole process. This low-level automated unit has few degrees of freedom and no complex controller is required. For example a rejected pouch can be pushed off the conveyor by a simple pneumatic cylinder activated by a single digital signal. A vibratory conveyor can be used to align the pouches. The stretching of the pouch can be performed by a dedicated device built for the specific pouch size inspected.

The fixed-automation devices are always useful to supplement flexible-automation devices present in the workcell. Such devices can, in many cases, release the generic robots (flexible-automated) from doing easy tasks, improving significantly the overall process time. Applications can be optimized in terms of cost and performance if an adequate balance between soft and hard automation is used when designing and implementing the system.

4.4 Hardware Models

The two hardware models are now given. The purpose is not to present the models up to the point of prototype design. The idea is to outline the (possible) components needed to build a workcell in terms of robotic requirements. This is why this section does not include detailed designs. While implementation of any of the models discussed here is not finalized, all the parameters are kept between reasonable ranges. Also, a discussion on the possible driver devices required to control the robot workcell is included.

A schematic representation of the vision system is outlined in Figure 21 for the case of a single side inspection. In general, the vision system should not interfere directly with the pouch manipulation, although any manipulator design must satisfy all the requirements demanded by the vision system.

As it will be shown in the next section, the vision system and the image processing are the most critical parts of the STP11. The vision system controller is also described in Section 5. This particular device is considered independently of the manipulator controllers to facilitate the analyses. However, it is clear that certain signals must couple the two systems to allow a synchronized operation.

The two conceptual models designed specifically to deal with the pouch manipulation required for

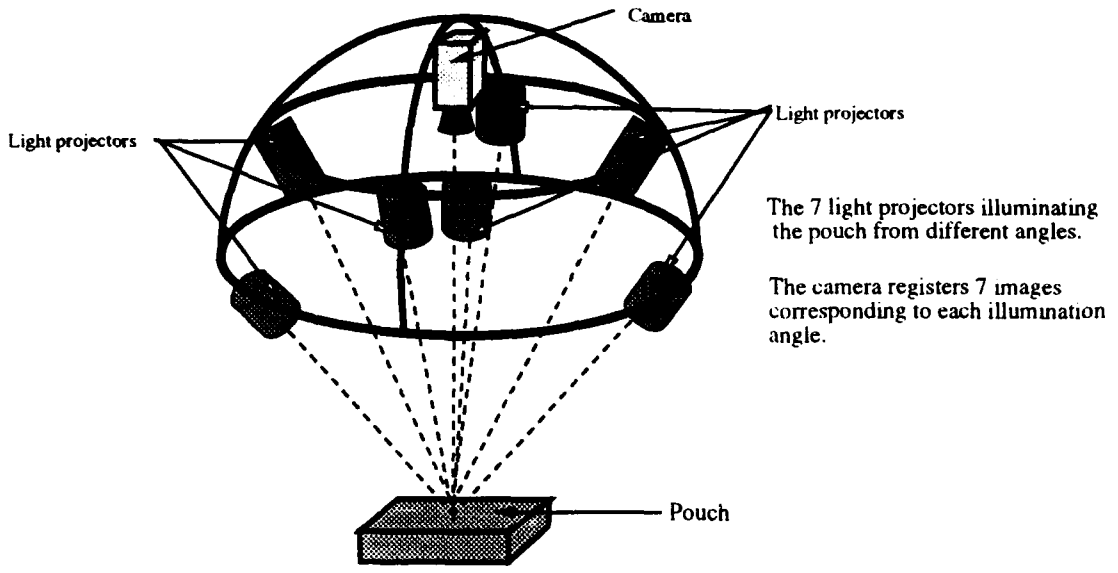


Figure 21: Vision system outline

STP11. are now presented. The models make used of the concepts of flexible and hard automation. It will be clear how the performance and implementation are related with the type of automation.

4.4.1 Model I

General Description This is a more general case of the example cited in 4.3. Here two robot manipulators are used in the pre-inspection and the post- inspection zones. The inspection manipulator is a fixed-automation device. The model is presented in Figure 22 and described in terms of the three manipulation zones as follows:

- *Pre-inspection manipulator* The first robot grasps the pouch from a feeder conveyor and transfers it to the inspection manipulator. This robot is a four degrees of freedom manipulator. Three degrees are used to position the end effector on top of the pouch, the fourth degree is used to compensate the possible misalignment of the incoming pouch. The pouches are grasped by means of a vacuum gripper end-effector.

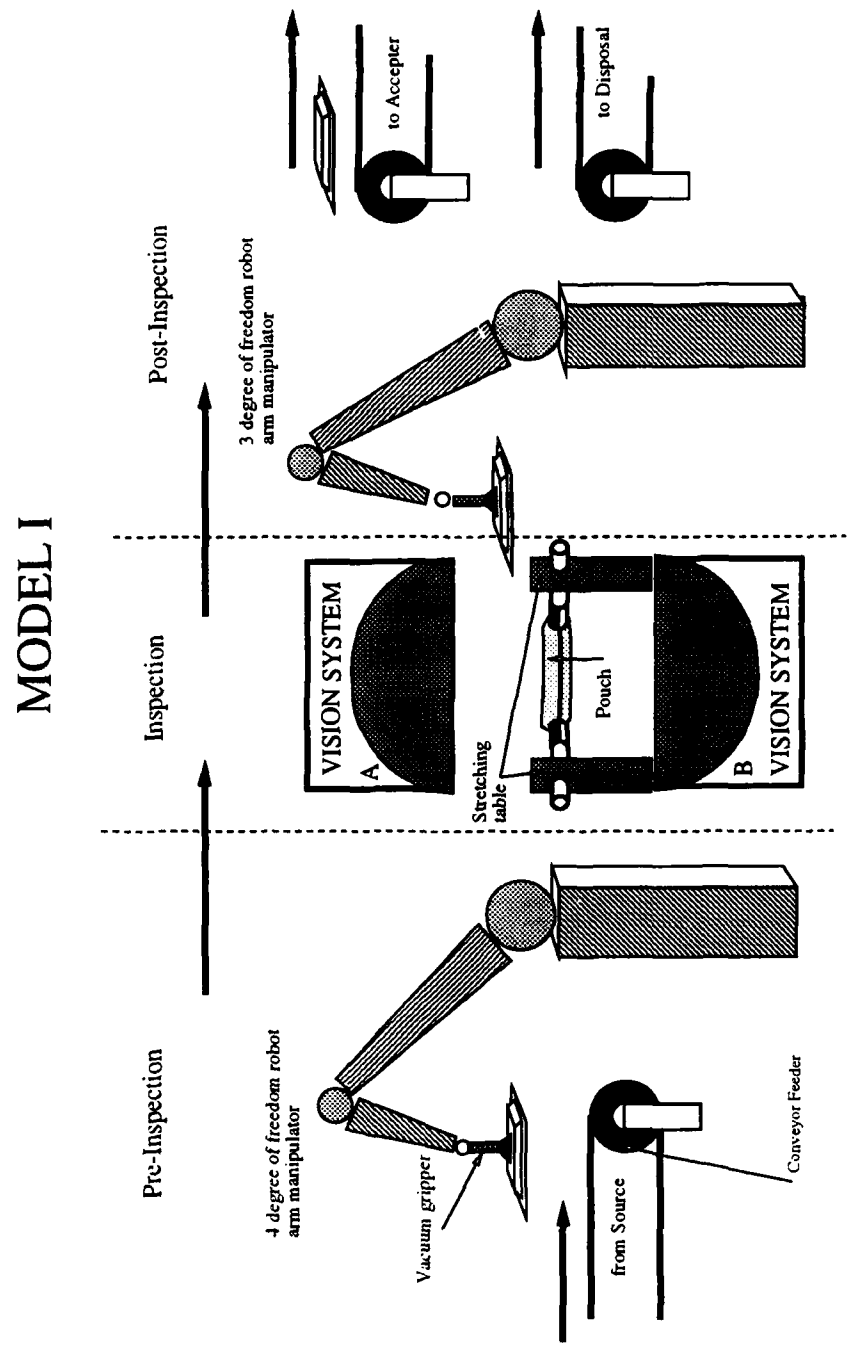


Figure 22: Model I

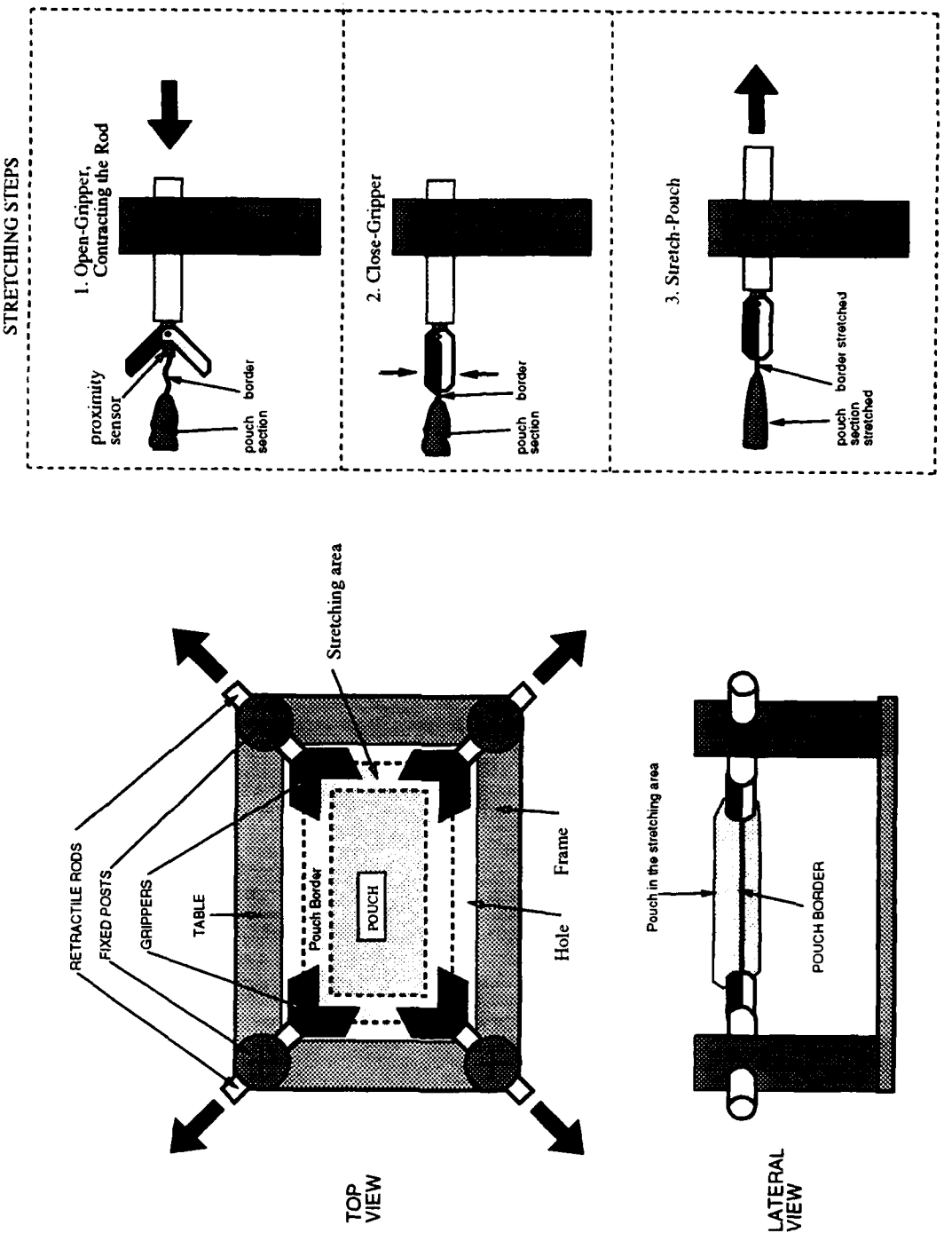


Figure 23: Inspection manipulator: The stretching table

- *Inspection Manipulator* A special fixed-automation device is used here. It is called a *stretching table* and is shown in Figure 23. It consists of four retractile rods —prismatic joints— each one having a pneumatic gripper at its end. The process of stretching the pouch is as follows:
  - *Step 1*: The pouch is brought by the *pre-inspection* manipulator to the stretching position in the middle of the four retractile rods.
  - *Step 2*: The retractile rods approach the pouch at the corners until the proximity sensor installed inside the gripper detects the pouch. At this moment the four grippers are closed holding steady the pouches by the sealed areas at the corners. The *pre-inspection* manipulator releases the pouch and moves away.
  - *Step 3*: The rods retract stretching out the pouch. At this moment the images are registered in both Vision Systems which take simultaneous images of both faces of the pouch.
  - *Step 4*: The *post-inspection* manipulator comes into the inspection zone and holds the pouch with a vacuum gripper while the stretching table opens the grippers releasing the pouch.
- *Post-inspection Manipulator* This is a 3 degrees of freedom manipulator that takes the pouch from the stretching table (step 4 of stretching process) and puts it into outgoing conveyors. The orientation of the pouch is not relevant in this manipulation zone. Two conveyors are available, one for accepted and one for rejected pouches. For this robot to be able to decide where to place the pouch the machine vision system must have an evaluation on the pouch's quality before the robot finishes its trajectory. Otherwise the robot will dump the pouches in a single conveyor and another device (fixed-automation) removes the defective pouches later. This latter procedure is attempted to avoid keeping the post-inspection manipulator in a hold.

The stretching table (ST) is a typical example of fixed-automation. Such a device must be designed and implemented for the specific needs of the workcell. Even though there are many ways to design the ST, only one is presented here.

The retractile rods can be driven by pneumatic cylinders. Figure 24 shows a schematic outline of the proposed device. A single piston **A** pushes the rod **R** forward towards the pouch . A spring **S** is used to retract the rod once the pneumatic piston **A** has been decompress. An extra pneumatic cylinder **B** is used to quickly stop the rod's forward motion. The rod is stopped once the pouch has been detected by the gripper sensor **D**. Break **B** is used because of air compressibility which makes it very difficult to control the position of pneumatic actuators **A**. Servo positioner devices (e.g. DC serve motors) that can perform such a precise positioning control are more complex and expensive.

The sensor (or sensors) **D** located at the gripper detects the corner of the pouch and can be implemented by an infrared or ultrasonic proximity detector. The function of this sensor is to determine when the retractile rod should stop so that the grasping is safe. Without these sensors the gripper could miss the corner or could destroy the pouch.

If the conditions are such that the pouch is alway placed in the same exact position, the stretching table can be simplified. This can happen if the position and orientation of the pouch at the pre-inspection zone are completely determined. In this case, the pre-inspection manipulator can place the pouch precisely in the middle of the stretching table aligned with it. For this condition the stretching table does not need either proximity sensors or breaking systems.

The gripper actuator is a double action pneumatic actuator **C**. The particular design shown in Figure 24 uses a double-side sliding gear mechanism. As the cylinder road slides, it drags with it two geared wheels attached to each finger. A double action cylinder has been chosen to guarantee the complete release of the pouch before the retractile rod retracts. If for any reason, one of the four grippers is still holding the pouch while the rods are retracted, the pouch can be torn or pulled out of the vacuum grippers.

Implementation and Control The particular elements that can be used for implementation are now discussed. The suggested items are not necessarily the best, they are just examples of specific available products suitable to perform a required task, or required to build a specific fixed-automation device. Also,

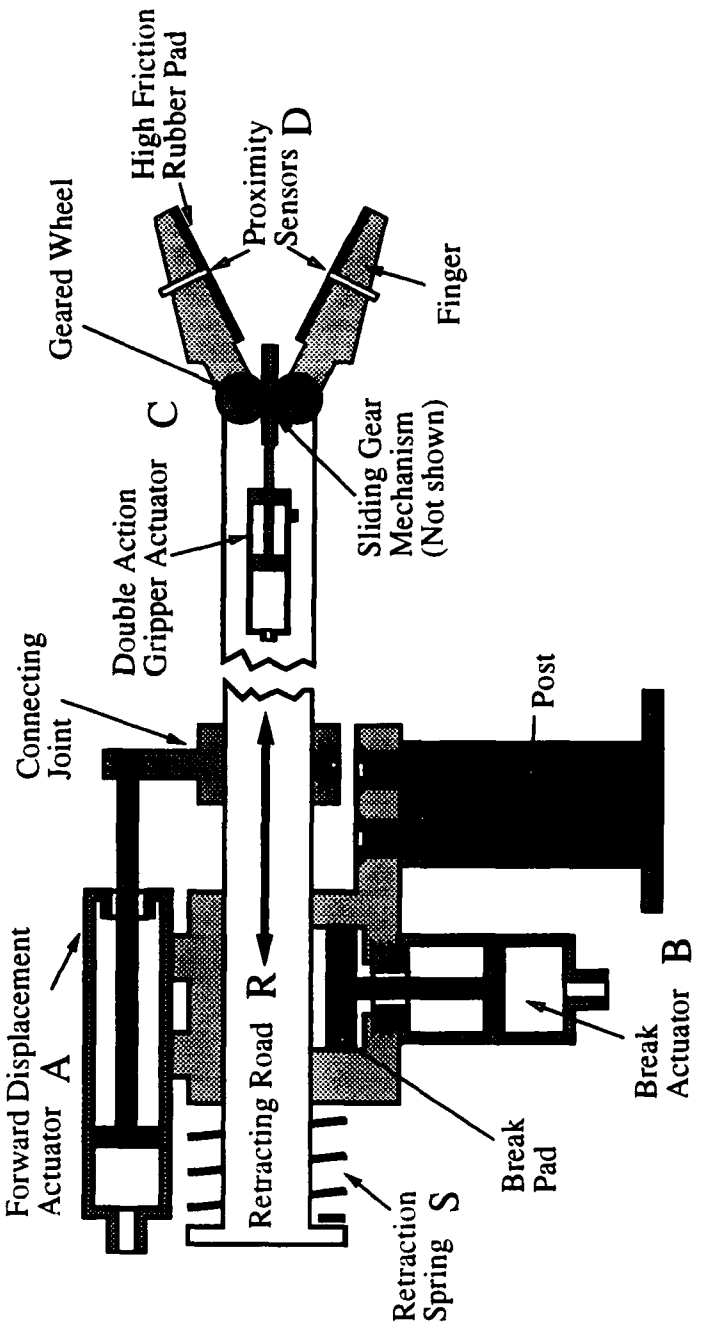


Figure 24: The retractile rod of the stretching table

a brief discussion about the electronic devices (interface, computer, etc) required to control the robot workcell is included. If possible, an outline of the overall system architecture is presented.

Many commercially available robot manipulators can be used to implement the pre-inspection and post-inspection robots. For example the popular industrial IBM 7535 Scara is adequate. Four reasons support this choice:

- The required work envelop is small.
- The required speed is high.
- The arm geometry is adequate for the task.
- The required payload is very low.

However, many other robots similar to the Scara can be used. For instance, the AR-H250 made by Hirata Industrial Machineries Co. has the Scara geometry but a lower payload and also a lower cost. The retail price of one of these units including their controllers are between $20,000 and $40,000.

A commercial industrial robot of this type comes with its own controller. This fact eases their application because such devices are already loaded with the programs needed to optimally utilize the manipulator. Those controllers often include a high-level language interpreter for easy programming.

One of the elements that, in general, is customized in any industrial robot is the end-effector (such a device is task-dependent). For the case of MRE pouches, it is advisable to use a vacuum gripper. The reason is that the very soft material of the pouch easily squeezes under low pressure. For this reason its exact shape is not well defined making it difficult for a finger-like end-effector to grasp with precision and stability. On the other hand, the pouch little weight (from 100g to 500g) allow the use of low vacuum levels for a safe grasping. The latter also guarantees that suction does not damage the pouch.

Many commercial vacuum grippers are available. For instance, the VG102-1 made by SENSOFLES/ASTEK is a good example of one such device. The price of this units is relatively low (around $500). This type of device includes its own vacuum pump which makes implementation a matter of installation. A wide range

of levels of vacuum can be chosen and the activation or inactivation of the vacuum gripper is controlled by a digital signal.

In general, the end-effectors do not require an extra controller. They are, in many cases, controlled by a few signals. Industrial robot controllers have several general purpose ports which can be programmed within manipulator program. In this way the generic end-effector becomes another extra degree of freedom, making it possible to include its behavior into the overall control strategy.

Once the robot and its controller have been installed, all that is needed is to program the required movement sequences using the high-level language supplied by the controller. A way to interface with the controller is via a host computer. All robot controllers have a host computer interface that allows communication with a general propose computer. This facilitates the simultaneous use of several robots in a single workcell. A host computer is used to monitor and command all the robots.

The host computer does not directly drive actuators as, shown in Figure 25. The host computer monitors the task execution programmed in each robot and it is able to initiate, change, stop or resume the activities of each one of them. A host computer ideally should interface with all the devices present in the workcell so that it can synchronize the activities. The host computer also interfaces with a human operator who has the ultimate control over the entire process.

Some of the devices connected to the host computer are external sensors. Sensors are placed in critical locations to directly monitor the state of the workcell. The information from the sensors allows the host to take action in real-time according to the workcell state. The feedback signals from the sensors are required in order to build an accurate and reliable workcell. Figure 26 shows a schematic representation of the feedback loop for the case of n robots.

Sensor information can also be sent to the robot controllers. That is the case of sensors that directly affect an specific robot. At the robot controller, such information can be used to take decision without consulting the host. For instance, a collision-detection device must communicate directly with the controller of the robot involved.

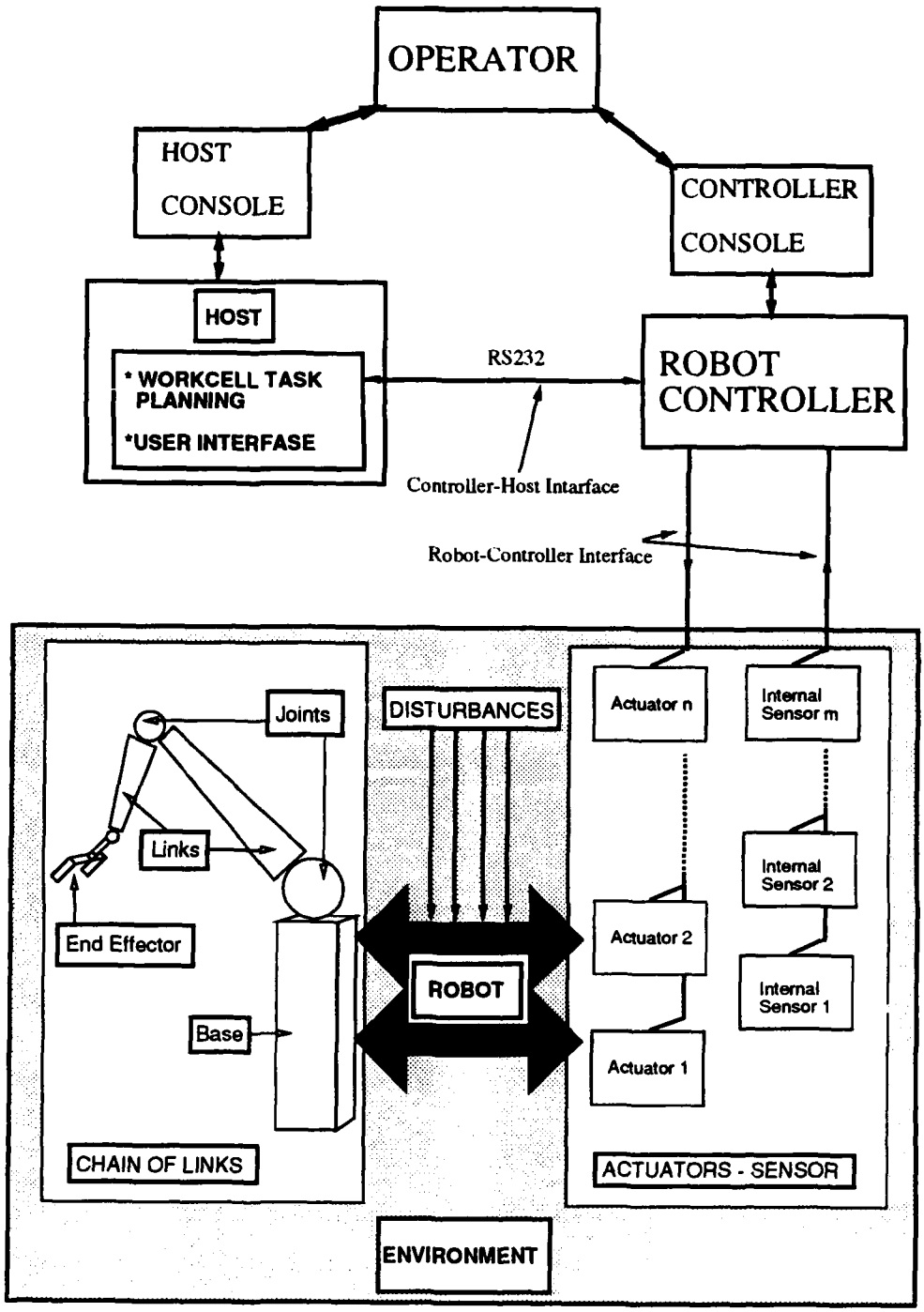


Figure 25: Robot manipulator control chain

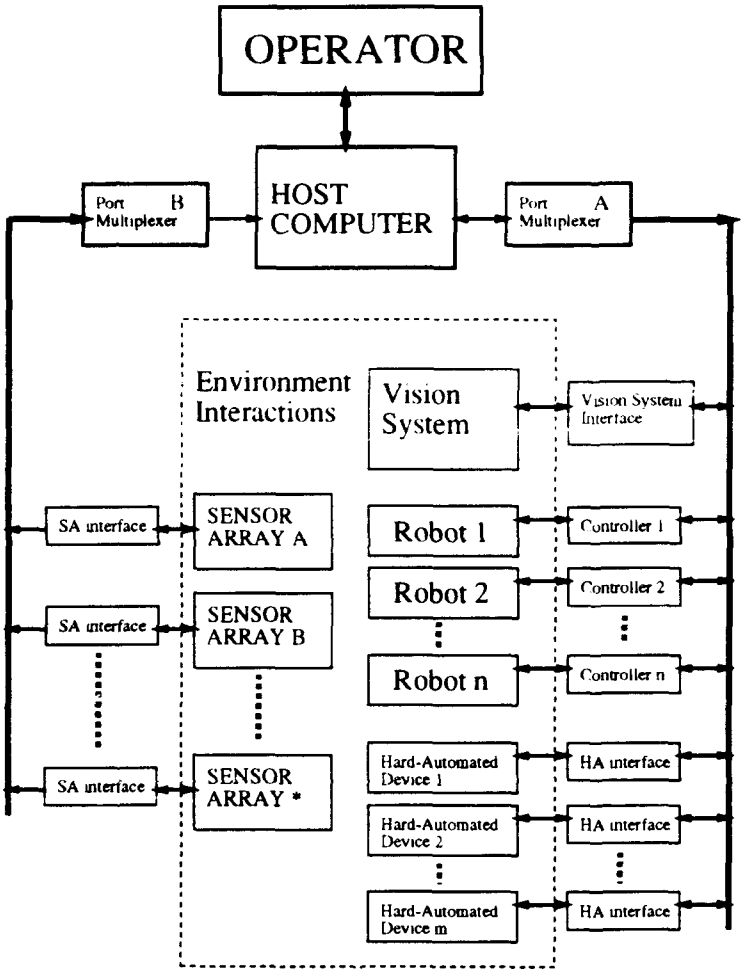


Figure 26: Control loop for robot workcell

On the other hand, since the I/O port of the host can be limited, the robot controller can be used as a general interface. That is, by encoding and sending data (together with robot data) through the controller communication port. Such a scheme reduces the number of cables and interfaces because the usual links between host-controllers are serial ports.

4.4.2 Model II

General Description The following model is based mostly on fixed-automation. However a robot manipulator described here may be used in other applications as well. It is considered a fixed omated

device because it has to be built for this specific task, and is not commercially available.

Model II uses a custom designed cartesian robot and a stretching table (described in 4.4.1) as shown in Figure 27. The robot has two simple arms, the first one **A** is the pre-inspection manipulator and second one **B** is the post-inspection manipulator. Each arm is equipped with a vacuum gripper end-effector. Because of the inspection cycle, the two arms may be driven by the same actuator. However as it will be shown, larger flexibility is reached if they are made independent.

The arms move along a common horizontal rail above the conveyor. Even though the rail is parallel with the conveyor, it is shifted to the side so that it does not interfere with the vision system.

The cycle to inspect a single pouch is as follows:

- While arm **A** is grasping a new pouch (from the feeder conveyor) to be inspected, arm **B** is removing an inspected pouch from the stretching table.
- Then arm **A** moves so that its end-effector is on top of the stretching table and arm **B** moves so that it is on top of the outgoing conveyor (towards the right of the figure).
- Arm **A** deposits the new pouch on the stretching table while arm **B** releases the inspected pouch in the outgoing conveyors. If the inspected pouch must be discarded, arm **B** releases the pouch on top of the rejected conveyor before it reaches the accepted conveyor (see Figure 27).
- Arm **A** moves away from the stretching table (towards the left of the figure). The vision system takes the required images.
- Arm **B** moves back so that its gripper is on top of the stretching table. If the inspection process requires more time, then arm **B** stops so that it does not interfere with the vision system.

Components and Implementation The stretching table used in Model II is the same as the one used in Model I.

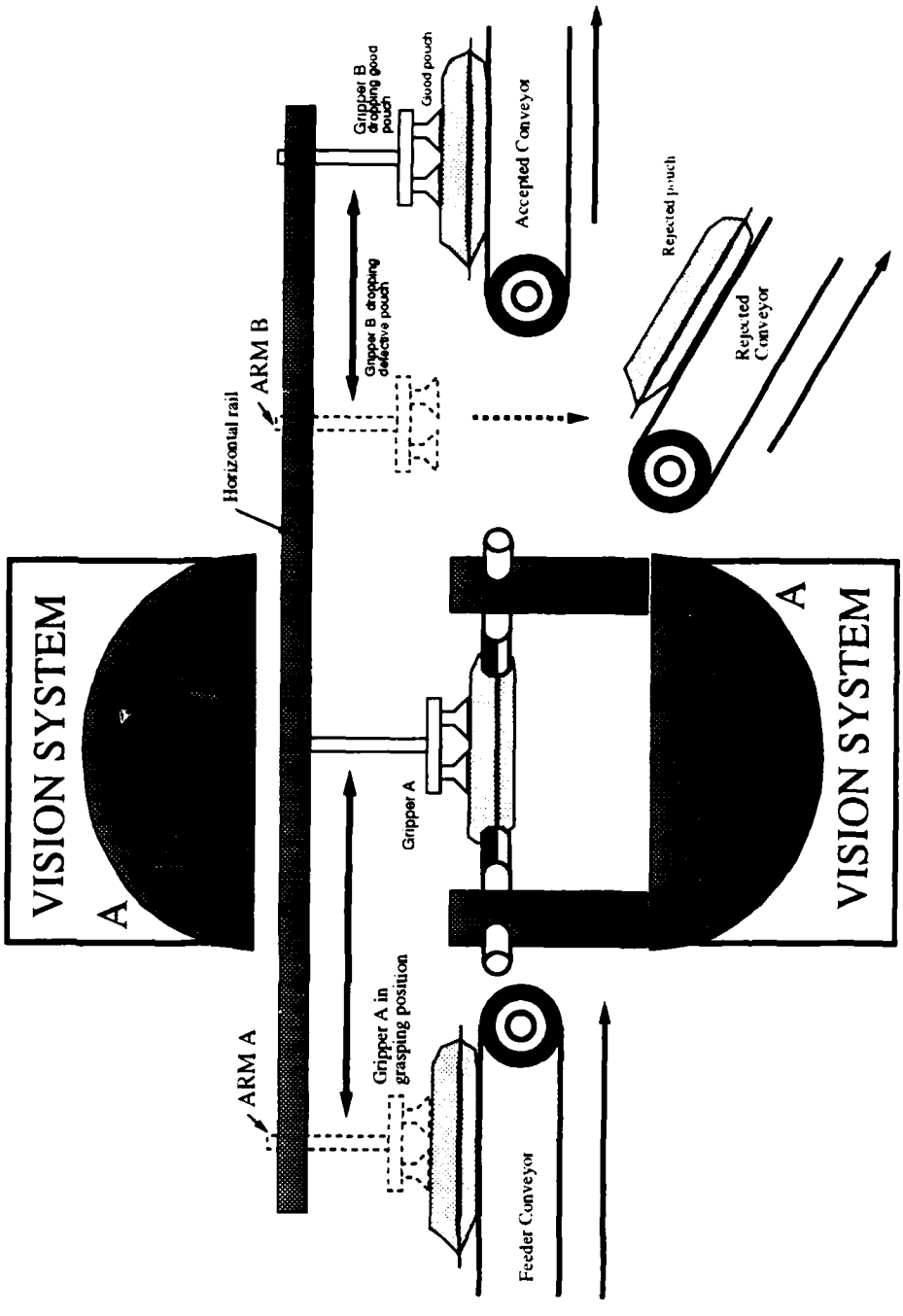


Figure 27: Model II

Arm **A** has to move up and down to pick the pouch from the conveyor without interfering with the pouch's horizontal displacement (tracking). On the other hand, arm **B** must move up and down to remove the pouch from the stretching table to avoid colliding with the pouch laterally. This horizontal movement of the arms can take place simultaneously and does not require any precise displacement control. Hence it can be implemented with pneumatic cylinders. The horizontal movement along the rail can be implemented by means of an AC or DC servo motor.

The position of the pouch (on the feeder conveyor) can be detected with a mechanical micro-switch or an infra-red light beam sensor(very low cost methods). Assuming that the pouches are always aligned on the feeder conveyor, arm **A** does not need additional degrees of freedom to compensate for deviations (i.e. rotation about the perpendicular axis to compensate the pouch orientation or translation to compensate for displaced positions on the conveyor). In this case, the mentioned sensors are enough to guarantee that arm **A** will descend in the middle of the incoming pouch. As it was said earlier, this can simplify the stretching table design as well.

It may be possible to synchronize the whole process so that both arms can be displaced simultaneously with the same actuator. However, such a coupled system makes it difficult to optimize the control and to compensate for random errors, e.g. arm **A** looses its pouch and it has to go back to feeder conveyor while arm **B** should continue its normal activity.

Some of the possible actuators that can be used to implement the movement are:

- A DC or AC motors capable of moving both grippers at reasonable speed (2 feet/sec). The price of this unit varies through a wide range, from \$100 to \$1000. A reasonable motor is a DC servo-motor TTB2-2933-3020-HA made by INLAND Inc. It has 1.5 HP, 3.4Lb.-Ft torque, brake and motor-tach. The cost of this unit is \$400. It requires a servo amplifier that costs \$800.

- Two pneumatic cylinders to move the gripper up and down. A single action cylinder ESN-1-2 made by FESTO Inc. has 1in bore and 2in stroke and costs \$70.

- The vacuum grippers are the same ones used in Model I.

As all the automatic systems described here, Model II requires a computer controller. For custom designed robots there are generic controllers. Such devices are microcomputers based on standard micro-controller or micro-processor chips. They can be accessed by a host computer or they may have their own console. In either case, programs must be written to generate a library of routines specialized in controlling the particular manipulator.

A total cost of Model II is around $2000 for the complete unit. It is important to note that the air compressor and the computer controller can be shared among several of these units, reducing the costs. However, the detailed design and its prototyping (for testing) add extra costs, amounts that are beyond the scope of analyses of this research.

4.5 Conclusions

The price of an automatic workcell is a function of the automation level. If fixed-automation units are used, the price is low (in the range of $10,000 to $30,000). However, the reliability may be low too compared to a higher automation level system. Such systems are implemented by means of flexible-automated units like robot arm manipulators. Systems based on flexible-automation are expensive (in the range of $30,000 to $80,000 or more!). Nevertheless, these systems are very reliable and flexible.

Custom designed systems can be used. These systems are a mixture of flexible and fixed-automation (for instance, special designed arm manipulators). For these systems the overall cost decreases but the designing and prototyping of the custom designed robots adds costs whose analyses are not considered in this report.

Making use of currently used machinery can reduce the cost and complexity of the workcell implementation. As mentioned in this section, the precise grasping of a random oriented and positioned pouch increases the degree of complexity of the robotic system; increasing in this way the implementation cost. But for instance, if an inspection workcell is coupled directly to the current filler machine, there is no need

for grasping pouches randomly from a conveyor. These methods make use of whatever mechanisms the filler machines use to hold steady the empty pouch, fill it and seal it. This particular solution requires further analysis of current used machinery. Such analysis is suggested as future work.

5 The Vision System

5.1 Introduction

This Section presents a discussion of the initial vision system proposed by the STP11 research team. It contains also a discussion of the proposed modifications. This Section describes the first vision inspection prototype. Based on this experience some possible improvements of the system and some alternative solutions will be presented in the final part of this section.

The vision system is the most critical part of the quality control of MRE pouches. 100% of the inline inspection is visual. In current manual plants, human operators look at the pouch for several seconds and based on their experience (and training) determine whether or not a given pouch is a defective one. The mission of machine vision is to develop a system capable of replacing the human inspector in this task. Such a system must be as reliable as the human operator and must be able to perform the inspection task in a reasonable time.

The objective is not to develop a system that can perform the inspection in less time than a human inspector can. It has been shown in Section 1 how the overall plant topology can be designed to compensate for a slower robot based workcell. However the reliability of the machine vision is very important. Here reliability means the success rate of the system, in other words, the percentage of time that a machine vision-based inspector is capable of successfully determining the quality of the pouch.

The performance of the machine vision inspector affects in such a way the overall plant production quality, that it can not be compensated with simple topological or parametric changes. The MRE defects were classified into four types, each of them with several classes as follows:

Type 1 - Critical

a) swollen pouch

b) tear, cut, hole, or abrasion/silver

c) fold over wrinkle

d) entrapped matter in seal

e) delamination

f) leakage(open seal, short seal)

Type 2 - Major A

a) unclean pouch

b) missing or incorrect labeling

c) impression of seal area

d) less than 3/16" tear notch seal area

Type 3 - Major B

a) color change after thermal processing

b) incorrect heat seal dimensions (minimum 1/16" seal width)

c) dislocated closure seal (< 1" from top of pouch to bottom of seal)

d) incorrect pouch external dimensions (4.75"x8.125")

Type 4 - Minor

a) missing or misshapen tear notch

b) dislocated tear notch

c) incorrect tear notch depth

The machine vision technique discussed in this section deals only with the defects that expose the pouch's metal coating, such as tear, cut, hole, abrasion, delamination. As it will be clear later, some types of unclean or swollen pouches are also detectable by the vision technique presented here.

5.2 Background Theory

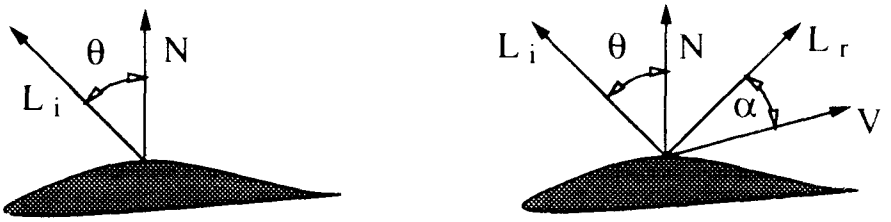
The detection of defects by machine vision techniques implies the analyses of images taken from the pouches. Such images are made possible because pouches (as most other objects) reflect visible light. Reflection and refraction are, indeed, very complicated processes whose thorough analysis involves the knowledge of the electromagnetic interaction of photons and matter. Fortunately, for the model under consideration, such analyses are not required.

The behavior of the phenomenon involved here can be modeled with the help of simple empirical descriptions. Many of such descriptions do not necessarily correspond to a real physical phenomenon, but for the scope of this study they are appropriate.

We start by introducing simple definitions on which the defect detection model is based.

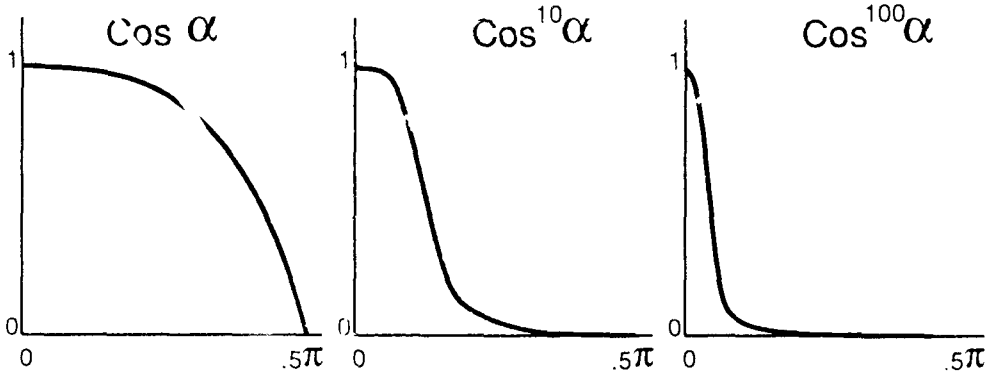
Diffuse-Reflection: Also called Lambertian reflection. Dull, matte surfaces like chalk exhibit *diffuse reflection.* Such a surface appears equally bright from almost all view points. This is because these surfaces reflect light with (almost) the same intensity in all directions. The reflected intensity is a function of the angle θ between the normal to the surface N and the direction of the incident light beam L_i (See Figure 28a). For a single point–light–source the Lambertian diffuse-reflection equation is given by:

$$I_d = I_i k_d cos(\theta) \tag{7}$$



a) Diffuse

b) Specular



c)

Figure 28: Reflection models

Where k_d is the *diffusion-reflection coefficient* which is a constant between 0 and 1. k_d depends upon the intrinsic micro-structure of the material surface.

For the case of spectral (light frequency) analysis, I_d can be decomposed into the primary color components as follows:

$$\begin{aligned} I_d^{red} &= I_i k_d^{red} cos(\theta), \\ I_d^{green} &= I_i k_d^{green} cos(\theta), \\ I_d^{blue} &= I_i k_d^{blue} cos(\theta); \end{aligned} \tag{8}$$

where k_d^{color} is the *diffusion-reflection coefficient* corresponding to each color.

If the whole visible spectrum is to be considered, equation 7 can be extrapolated to the continuous wave length interval.

$$I_d(\lambda, \theta) = I_i(\lambda) k_d(\lambda) cos(\theta), \tag{9}$$

where $k_d(\lambda)$ is the *diffusion-reflection coefficient* as a function of the wave length λ of the light source.

Specular Reflection: Any shiny surface (e.g a mirror) exhibits *specular reflections*. These particular reflections produce a highlight spot on the object surface. Such a reflection is a function of the view point, the light source location and the topology of the surface. The color of the specular reflection does not in general resemble the object's color, the light source color dominates the specular reflection. A perfect mirror will only reflect light on the direction L_r (See Figure 28b). A viewer will be able to observe the specular reflection if his/her vantage (view) point is on the line generated by L_r. In this case $\alpha = 0$, and the specular reflection is just the image of the light source.

A model for specular reflection was developed by Warnock [71] and later expanded by Phong [8](who gave his name to the model). The Phong model assumes that the maximum intensity of the specular reflection occurs when $\alpha = 0$ and falls very fast as α increases. The intensity decay is assumed to be proportional to $cos^n(\alpha)$, where n is the *specular-reflection exponent* of the material. The values of n vary between 1 and several hundreds. Large n provide a sharp focused highlight and small values correspond

to a broad gentle one. A perfect mirror cited above has $n = \infty$. Figure 28c shows several plots of $cos^n(\alpha)$.

The intensity of the specular reflected light given by Phong's model is:

$$I_s(\lambda, \theta, \alpha) = I_i(\lambda) W_s(\theta) cos^n \alpha \tag{10}$$

where $W_s(\theta)$ is the object's *specular-reflection function* which in general depends on the incident light angle θ.

Ambient Light: There is another component of the reflected intensity. The ambient-reflection which is due to the diffuse ambient light. This reflection is easily modeled by:

$$I = I_a k_a \tag{11}$$

where k_a is the object's intrinsic *ambient-reflection coefficient* and I_a is the ambient light intensity assumed to be constant for all objects regardless of position or direction.

Total Reflection: The overall reflection intensity is obtained from the superposition of the diffuse-reflection, specular-reflection and ambient light intensities given by equation 9, 10 and 11 as follows:

$$I_r(\lambda, \theta, \alpha) = I_a k_a + I_i(\lambda)[k_d(\lambda) cos(\theta) + W(\theta) cos^n(\alpha)] \tag{12}$$

I_r is the reflected intensity which is a function of the variables λ, θ and α

The environment lighting condition given by the ambient light term $I_a k_a$ introduces a constant term that will be irrelevant for the analysis of the model. Hence, such term will be dropped from now on.

The *reflectance function* is defined as:

$$\Phi(\lambda, \theta, \alpha) = [k_d(\lambda) cos(\theta) + W(\theta) cos^n(\alpha)]. \tag{13}$$

so that the reflected intensity is given by:

$$I_r(\lambda, \theta, \alpha) = I_i(\lambda) \Phi(\lambda, \theta, \alpha), \tag{14}$$

The *reflectance function* can be divided into two main components:

$$\Phi(\lambda, \theta, \alpha) = \Phi_d(\lambda, \theta) + \Phi_s(\lambda, \theta, \alpha), \tag{15}$$

where $\Phi_d(\lambda, \theta, \alpha)$ is the *diffuse-reflection function* given by:

$$\Phi_d(\lambda, \theta) = k_d(\lambda)cos(\theta), \tag{16}$$

and $\Phi_s(\lambda, \theta, \alpha)$ is the *specular-reflection function* given by:

$$\Phi_s(\lambda, \theta, \alpha) = W(\theta)cos^n(\alpha). \tag{17}$$

The reflected intensity can be finally written as the superposition of the two types of reflection.

$$I_r(\lambda, \theta, \alpha) = I_i(\lambda, \theta) + I_s(\lambda, \theta, \alpha) \tag{18}$$

where the diffuse-reflection intensity is

$$I_d(\lambda, \theta) = I_r(\lambda)\Phi_d(\lambda, \theta) = I_r(\lambda)k_d(\lambda)cos(\theta), \tag{19}$$

and the specular-reflection intensity is given by

$$I_s(\lambda, \theta, \alpha) = I_r(\lambda)\Phi_s(\lambda, \theta, \alpha) = I_r(\lambda)W(\theta)cos^n(\alpha). \tag{20}$$

5.3 The Defect Detection Model

The defect detection model discussed here integrates two techniques to isolate defects from the normal surface. For the purposes of this report, these techniques are given the names *color-subtractive* scheme and *multiple-illumination* scheme.

Figure 29 shows a schematic of the *color-subtraction* scheme. The system consists of one single light source (white), a color filter, a pouch and a camera. The image registered by the camera is considered to be made of reflected light from the light-source by the pouch surface.

Without the filter (white light illumination), the defect-free areas present diffuse-reflections (mostly of the pouch color: green) and specular-reflections (of the same color as the light source: white). Specular-reflections produce shiny spots that are function of the surface irregularities, the light source and camera position.

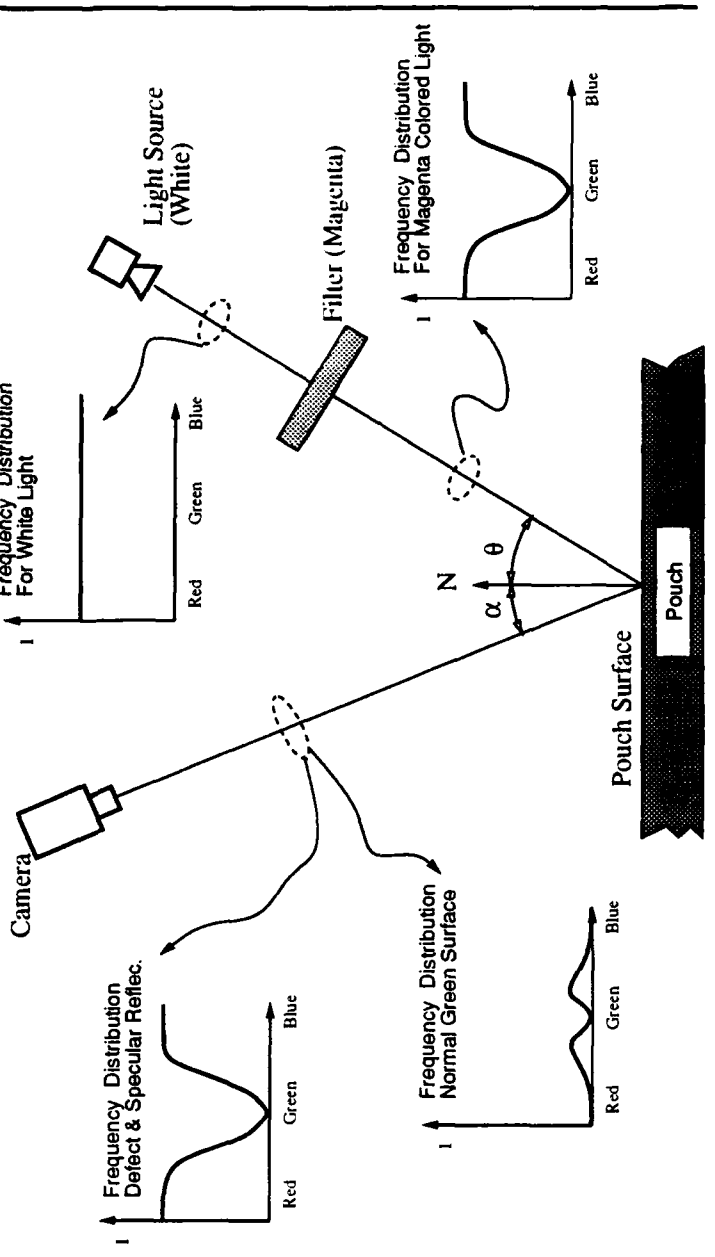


Figure 29: Defect detection system: Filtered light-source

Defective regions of the pouch contribute to the image with mostly diffuse-reflections and occasionally (for very particular conditions) specular-reflections. In any case, defective regions produce reflections of the same color as the light source. That is because the defective areas are assumed to be regions where the metal coating is exposed.

The color-subtractive scheme consists of illuminating the pouch with the colored light so that the diffuse-reflected green light of the defect-free region is almost completely suppressed from the image. If the light is filtered so that the incident colored light lacks the green component, the defect-free areas will absorb most of the incident light. This is made possible by using a magenta colored filter.

From the mathematical point of view the color-subtractive scheme is easily explained. In equation 19 the color distribution of the incident light $I_i(\lambda)$ is minimized on the values of λ where the *diffuse-reflection function* $\Phi_d(\lambda, \theta)$ of the material is maximum. That is the region of the spectrum corresponding to the green color.

Since the filter is not perfect (a perfect magenta filter has no green component at all) some residual diffuse-reflection is present in the image. However, the residual intensity is considerably lower than the intensity of defective regions.

Colored-subtractive schemes isolate and reduce the contribution of diffused-reflection from defect-free regions. However, the specular-reflection from these regions and the reflection from defects are of the same color as that of the incident light. Therefore, they cannot be isolated by a color-subtractive scheme. The goal of the multiple-illumination scheme is to separate these two components.

As described above, the specular-reflections given by equation 20 are functions of the surface shape and the relative position of the light source and camera. Assuming that the surface shape and the camera position remain constant,then, as the light source moves from one place to another, the relative position of the bright spot corresponding to specular-reflection moves as well.

If several images are registered, corresponding to different illumination configurations, it is possible by processing the complete set of images to separate the contribution of the specular-reflection. The main

issue in this technique is to use the fact that the pattern of specular-reflection is a function of the light source position. Since the placement of the light source does not change the object-image mapping (which is a function of the camera-object position), it is possible to easily correlate several images with different light conditions.

The bright spot in the image corresponding to defective regions appears in all the images. This is because the contribution of defective regions is mostly due to diffuse-reflections whose reflection functions change slowly for small values of θ as given by equation 19. On the other hand, the specular-reflection given by equation 20 is represented by a sharp function of the light source-camera phase angle α. From Figure 28 it is clear that $\Delta\theta = \Delta\alpha$ so that small changes in the light source position will produce large changes in the specular-reflection pattern while the diffuse-reflection pattern remains constant.

Figure 30 shows several images that demonstrate the principles discussed in this section. Figure 30.A is a gray level image of a pouch area. Figures 30.B and 30.C are two images of the same pouch with different illumination conditions. It is clear that the pattern of specular reflection is very different. Figure 30.D if the final result (after the "AND" process) for a set of seven images. In this case one defect was detected. It is clear that the defective area is present in all of the images.

5.4 Alternative System

Without going into many details it is possible to use the same reasoning presented in 5.2 to obtain an alternative system configuration. This alternative configuration was motivated by implementation problems, since it was found difficult to filter each single light source.

The principal implementation problem was that the light source (halogen light bulb) released large amounts of heat that no filter could withstand. This could be compensated by adding cooling systems, but implementation then would be more expensive and complex.

The same reflection model used earlier allows the derivation of an improved model which performs better than the initial one; it is also easier to implement and more immune to ambient light perturbation. Figure 31 shows a detection system based on a filtered camera with a white incident light source.

Figure 30: Detection system: Real images

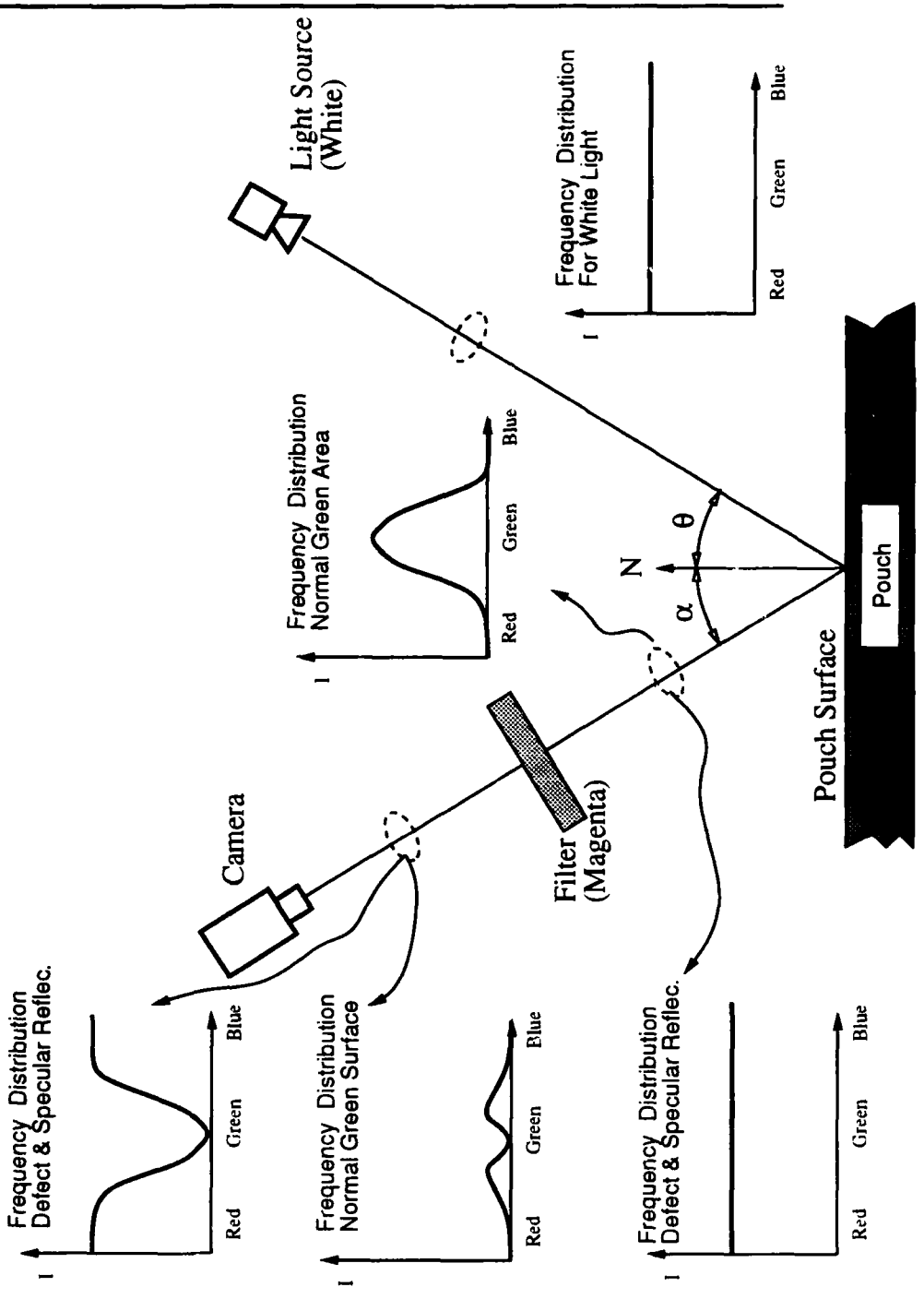


Figure 31: Defect detection system: Filtered camera

When using a white light source the light reflected by the pouch is heavily loaded with green components especially from those areas that are neither defects nor specular-reflections. Defective areas and specular-reflections reflect mostly light of the same color than the light source (white).

If a magenta filter is placed in front of the camera, after the reflected light has passed the filter, the green component of the pouch's normal surface is removed by the same color-subtractive scheme discussed earlier. The white light coming out of the defective region and the specular-reflection turn magenta because of the filter.

As shown in Figure 31 the two cases of defect-free and defective regions have the same color spectrum at the camera position as the one presented in Figure 29. So, this detection system (called filter-camera) is equivalent to the initial one, however, it is easier to implement since there is only one filter.

5.5 System Prototyping

5.5.1 The Lighting Structure

Given the analytical background, prototyping of the system resulted in a structure called "The Dome". Figure 33 shows a picture of the actual implementation. 'The Dome" is a hemisphere-like structure designed so that the camera is located at its zenith. Figure 32 shows a schematic representation of a seven light-source configuration. They are located along two meridian semicircles symmetrically placed at 15° of each side of the camera meridian. Along each of these two meridians 4 and 3 light sources, respectively, are uniformly distributed at 20° intervals.

The camera used in this implementation had a 512x512 CCD array. It was configured with a 16mm 1:1.4 lens, 10mm spacer and x2 extender. The camera was placed 30cm from the pouch with a diaphragm aperture of 1.4. For this set-up a image is a projection of a 3.2x3.2 cm region on the pouch, hence, the resolution of the system is 1/16mm x 1/16 mm per pixel. The light sources used were halogen 12V 50W. This type of light is a good source of white light.

The overall system configuration is shown in Figure 34. A PC computer drives the light array and

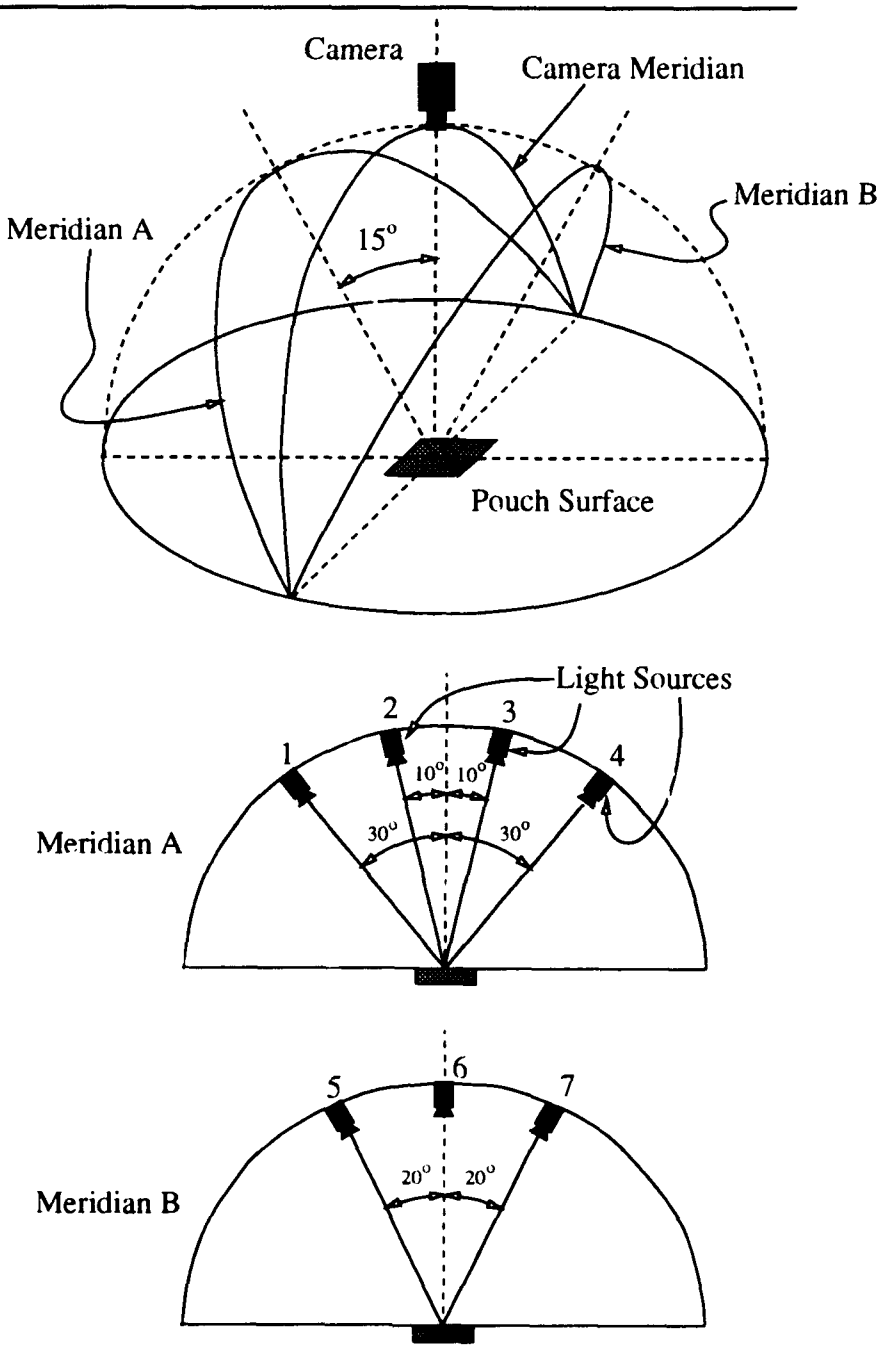


Figure 32: Light array distribution

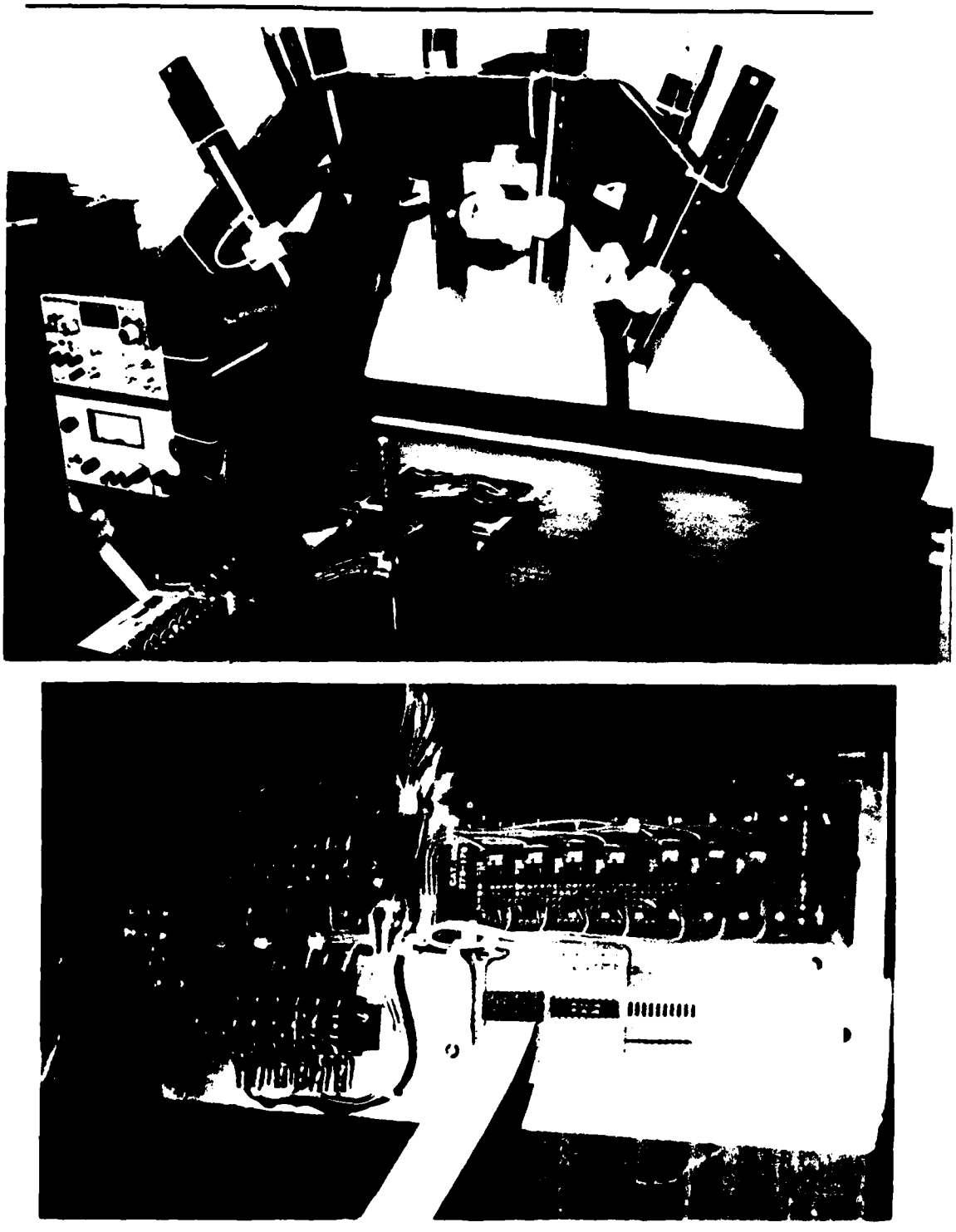


Figure 33: Prototype of the vision system

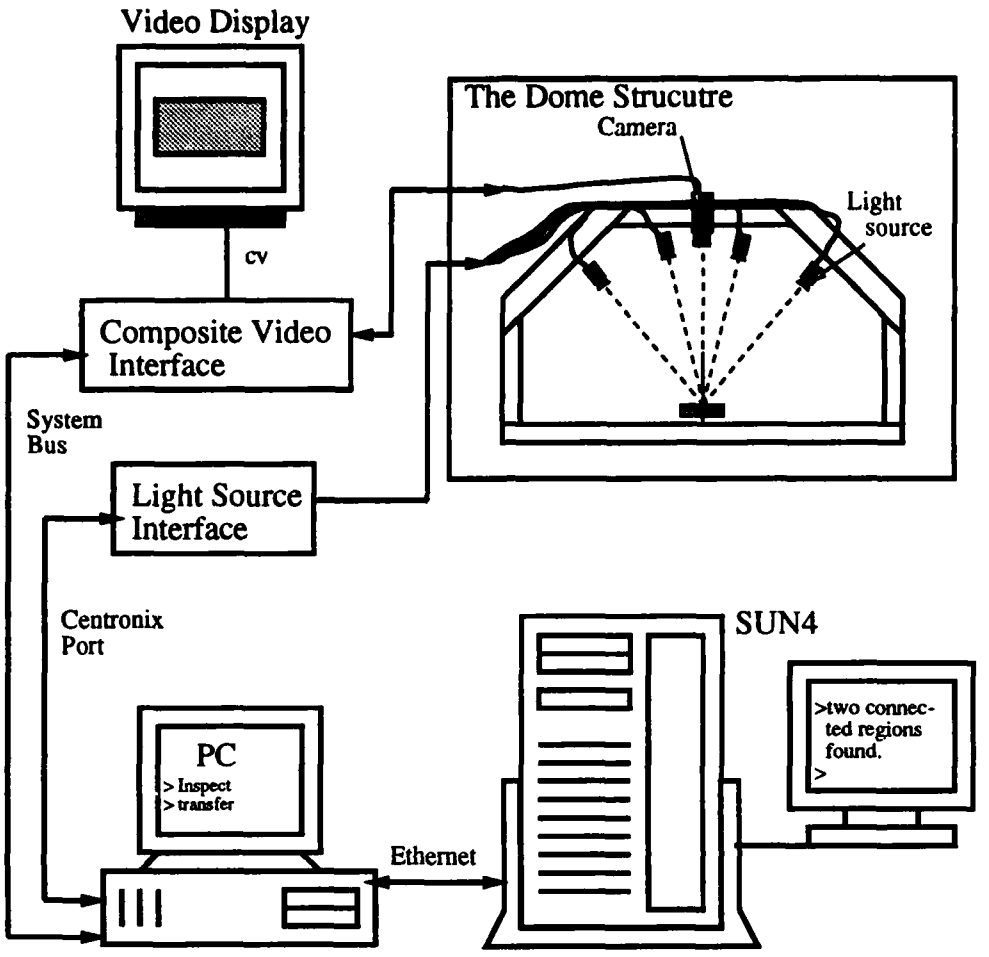


Figure 34: Machine vision system configuration

interfaces with the CCD camera controller. After the seven images are digitized by the PC the information is then transferred (via ethernet network) to a SUN/3 workstation which runs the image processing code.

A program running in the PC interfaces with the camera card controller to digitize images. Each image is initially stored in a 256KB RAM memory buffer and later transferred to a file in the HD. Before an image is taken, the program interfaces through the parallel port (Centronix protocol) with the specially designed light-array controller to switch each individual light on/off. In this way, seven images are registered each one with a different lighting direction.

A block diagram of the light-array interface controller is shown in Figure 35. It uses a 8-bit parallel-latch which uses the *Strobe* signal of the parallel port to latch a data byte. Each one of these signals drives

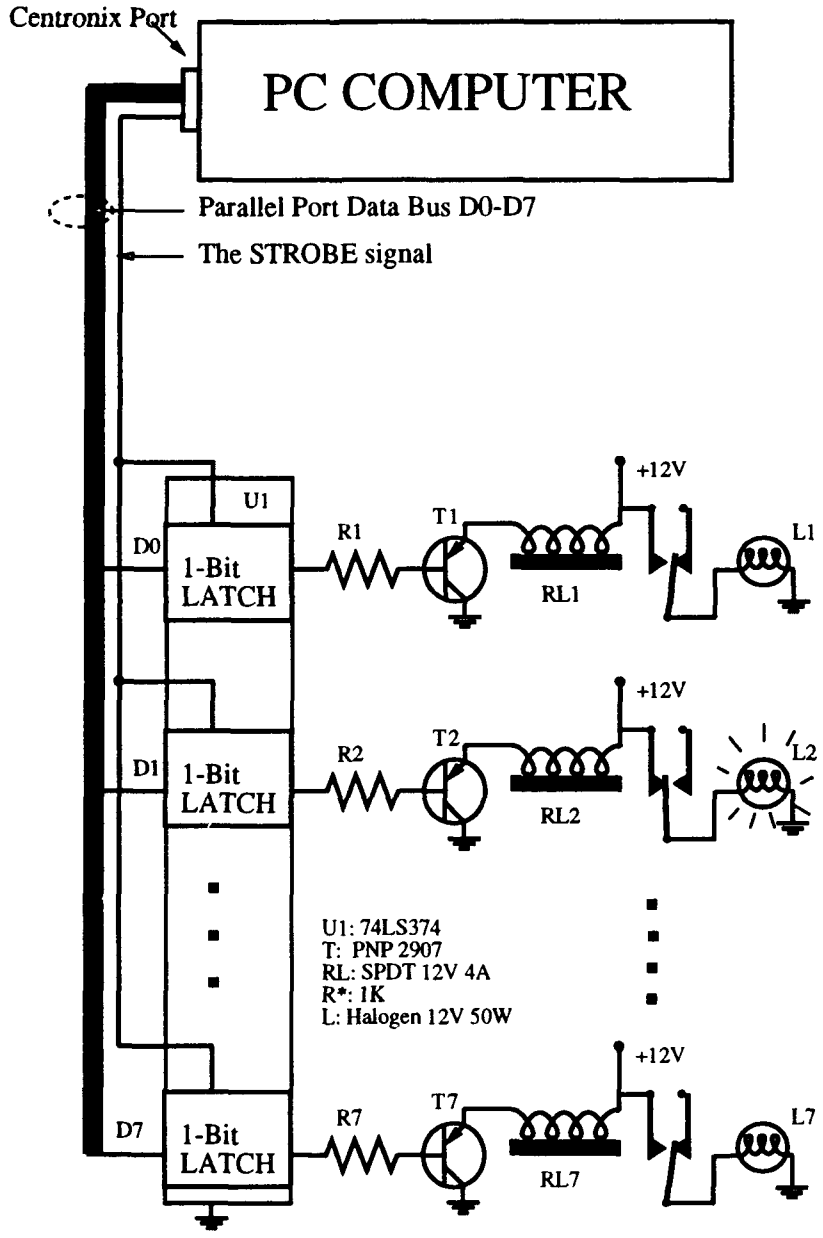


Figure 35: Light array interface

a PNP 2N2907 transistor which in turn is used to drive a power relay. The set of 7 relays drives the light bulbs directly. The function of this power amplifier chain is to convert the 5 volt 10 mA (50 mW) signal of the parallel port into a 12 volt 4A (50 W) required for the halogen light bulbs.

5.5.2 The Image Processing

Image Segmentation: Once the N gray level images of the same pouch, under different light directions, have been obtained, they are thresholded (binarized). A local thresholding technique must be used since specular-reflection can, in many cases, be brighter than reflection from small defective regions. In these cases, defective regions can be driven to background levels if a global thresholding technique is used.

The segmentation software divides the image into a block of m x m pixels. Each block is binarized with its own local threshold value. To do this, the histogram for each block is constructed and smothered.

Image Integration: The set of thresholded images undertakes an *AND* process to determine whether a region is present in all the images. As it was discussed before, the source of such a region belongs to areas on the pouch that diffusely reflect light from every light source; and is most likely a defective area. An alternative result "*N-1 out of N*" is also computed so that the consistency of the *AND* result can be evaluated. The "*N-1 out of N*" is a process that determines the pixels that appear on N-1 images out of the N images.

5.6 Conclusions

The vision system is the most critical component of the MRE robot based inspection. It is important to design a system which is reliable and efficient. The prototype system was trained on one set of MRE pouches and tested on a second set of pouches. Both the training and test sets were 10 MRE pouches of Ham slices.

For the prototype developed, a 5% error rate was obtained. That is, 5% of the true defects were missed; however, no defective pouches were missed. We found that most, if not all of the test defective pouches had more than one defect and in all cases at least one of the defects was detected. The technique proved

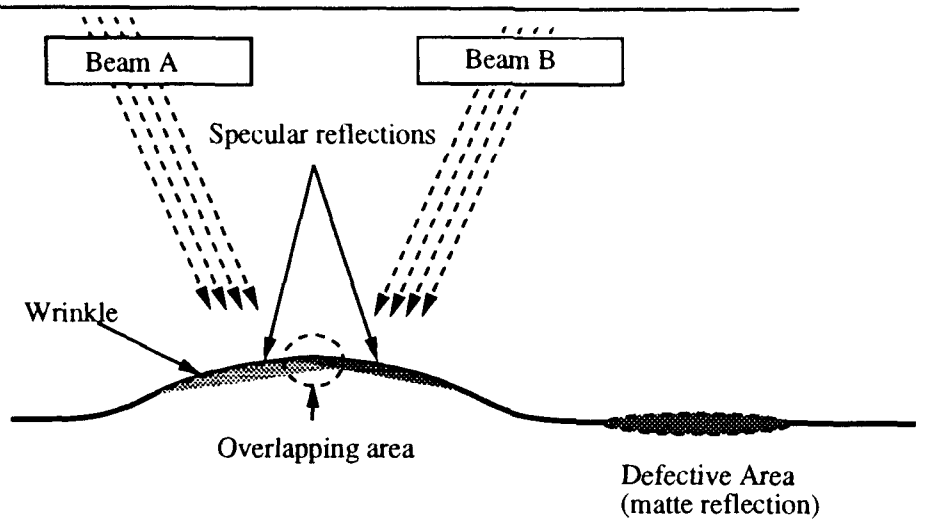
to work in general and the result is good for a first prototype system. Furthermore, none of the normal pouches were incorrectly classified as defective by the system.

Although this is promising, caution should be exercised and more extensive testing needs to be done on MRE pouches filled with other material that causes the surface not to be flat. As it was discussed before, the specular-reflection produced highlighted areas. For ideal mirror-like surfaces, these regions are concentrated in small areas. There positions change drastically as the light source moves. For regular surfaces, these reflections form large highlighted spots. If the light source position changes by a small amount (bright field of illumination) it is possible that there will exist overlapping regions that appear on all the images. This is because of the size of the specular-reflection; some regions present considerable specular reflection under all the set of light sources used by the multiple-illumination scheme.

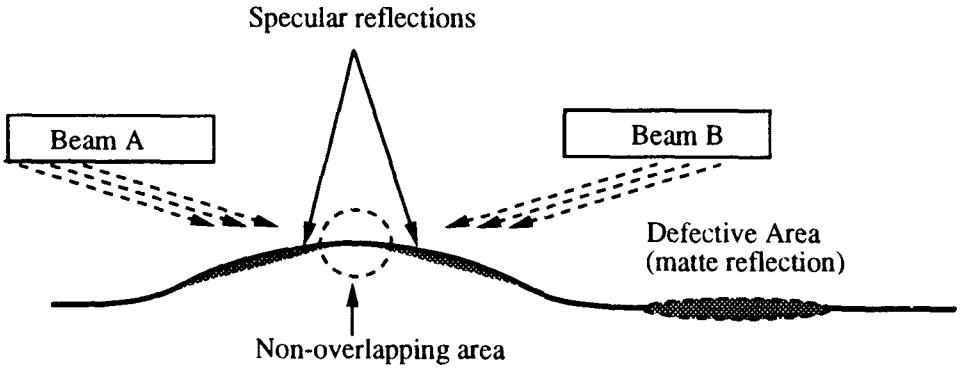
A schematic representation of this phenomenon is presented in Figure 36a. When a bright illumination field is used, it is possible that the two (or more) specular-reflection highlights have a region were they overlap. If the intensity of the region is such that several of the image's pixels are digitized to a foreground level, it is possible that a small set of pixels survive the *AND* process. In this case a defect-free area will be classified as a defective one.

The system can be improved if a dark field of illumination is used. Figure 36b shows such a situation for the case of two light sources. The two highlighted spots (corresponding to the two light sources) are separated enough so that the intensity of any overlapping region is small. In that case the probability of any overlapped region surviving the *AND* process is very small. Several experiments were performed where only four light sources distributed in a large illumination field were used (more than 50° with respect to the pouch normal). Even though the false-positive detection was decreased, two new problems appeared for this system.

If the incident light is placed at a large angle with respect to the surface normal, the diffuse-reflection component decreases as given by equation 5.3. Most of the light reflected out of a defective area is diffused, which means that for these illumination conditions the image corresponding to the defective areas will be



a) Narrow Illumination field



b)

Figure 36: Dark vs Bright illumination fields

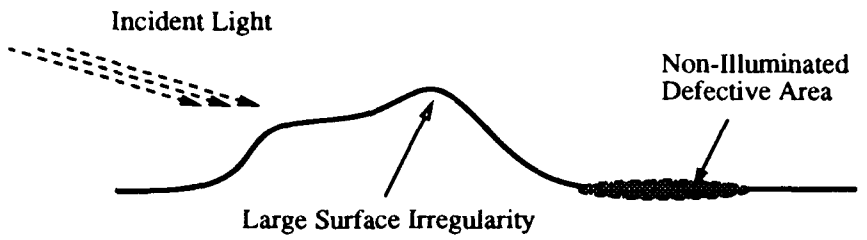


Figure 37: Hidden defective area

very dark (close to background level). In this case they are digitized to background and will not survive the *AND* process. In other words, defects will not be detected. The problem of low intensity reflections at large incident illumination angles can be easily solved if the intensity of the incident light source is increased for those large angles. The intensity can be increased by using more powerful light sources or just by placing the light source closer to the pouch.

Another problem that could appear is shown in Figure 5.6 where a defect region is hidden from the incident light due to a large surface irregularity. The problem becomes more notable when a larger field of illumination is used. A solution to this problem is to stretch out the pouch so that large wrinkles are flattened. The previous section described an automatic device that performs such an operation (the stretching table).

The image acquisition time is currently 7 seconds and the image processing, which is pipelined and parallel with pouch transport is 7 seconds. The processing time can be improved if parallel computer hardware is used. The segmentation of each of the image's frames can be carried in parallel. A dedicated set of image processors can be designated to implement such an operation. The images can be held in 256K RAM memory buffers attached to each processor to decrease the access time.

The thresholding of the first image can begin as soon as that image has been captured. While that first

processor is segmenting the first image, the camera can be loading image frames into the other processor buffers. It has been estimated that the total processing time will not exceed 7 seconds. Currently the prototype process takes several minutes, this is due to the data transfer among machines and the serial processing of the seven frames by the SUN3 workstation.

6 Economic Analysis

6.1 Economic Analysis of Robotic Workcells

A computer program for economic analysis of automated workcells (EARW) was developed by modifying an existing program written by Fang [25]. The description of this program is given in the following:

The procedures used in this analysis allow the summing of funds that occur at different times. Six indicators are produced for evaluating the economic performance of any user defined cases. They are:

1. Project Life (L),
2. Years of Operation (N),
3. Expected Return on Investment (EROI),
4. Return on Investment (ROI),
5. Cumulative Present Worth (CPW), and
6. Payback Period (YE).

They can also be used for comparison between different cases.

The amounts of funds for operation to be recorded are as follows:

1. Expense: fixed cost + operating cost
2. Revenue: the saving of reduced labor and value of eliminated manual system devices
3. Return: revenue - expense

Every record represents the situation of the flow of funds at the end of each year, starting from year 0 (which is the beginning of the project). The record of the zeroth year is:

1. Expense: the initial cost of robotic workcell (I)

2. Revenue: 0

3. return: -I

Taking the starting time of the project as "present time", the worth of the end of year i will be reflected to its "present worth" by dividing it by $(1 + \text{EROI})^i$. The sum of all the "present worth" from year 0 through N becomes the CPW.

EROI is an estimated value provided by the investor and ROI is the return rate on an investment over the lifetime of the project (i.e., when N=L). Payback period (YE) is calculated based on EROI, that is, the time required for achieving the expected return on investment. When EROI is zero, YE becomes the simple payback period (YS).

The following rules are used to interpret the result of this economic analysis:

1. If "N = L and CPW > 0" then "ROI > EROI"

2. If "N = L and CPW < 0" then "ROI < EROI"

3. If "N = L and CPW = 0" then "ROI = EROI"

4. If "N <= L and CPW = 0" then "YE = N with a return equal to EROI"

5. If "N <= L and CPW = 0 and EROI = 0" then "YS = N with 0 return"

The initial cost in this project is the cost of automated workcells. And, the revenue is labor savings, that is, savings of salary plus associated expenses. The economic analysis presented in this report is based on the above concepts.

The computer program (EARW) calculates a set of dependent variable values based on the user-defined independent variable values. EARW is a QuickBasic based program and its compiled version will run on most of DOS based microcomputers. The source code of EARW is listed in the Appendix. There are three categories of independent variables. The independent variables and their base-line values will be discussed later.

The cycle time is the average time required to finish one product. The task inefficiency indicates the percentage of unsuccessful cycles. The busy ratio represents the percentage of real working hours spent on production lines.

The dependent variables and their units are:

1. first year return in dollars
2. cumulative present worth in dollars
3. cumulative present worth in dollars per 1000 units of product
4. simple payback years, in years
5. expected payback years, in years
6. return on investment, in %

After entering the independent variable values, the program has three functions for economic analysis:

1. feasibility analysis,
2. parametric analysis, and
3. comparison of alternatives.

6.1.1 Feasibility Analysis

For a particular design, its economic feasibility will be evaluated using the manual operation as a reference. Therefore, the result always shows the comparison between the automated workcell and the manual operation based on the independent variables specified by the user. The result shows the interrelations among annual work volume, required daily productivity, number of working days, corresponding required number of workers and workcells, YE and ROI. The required daily productivity is the dominating factor which decides the number of human workers or automated workcells needed to perform the task.

6.1.2 Parametric Analysis

This part of program reveals the effects of variations of dependent variables on independent variables. During each program execution, one dependent variable and two independent variables can be displayed on a graph. The dependent variable is shown as the vertical axis, one of the independent variables is shown as the horizontal axis and the other one as the control variable. The user can select the variables and define their ranges. As a result, a family of curves are drawn showing the interdependence of the three variables involved.

By this analysis, the following are example questions which can be answered:

1. How low should the price of workcell be in order to obtain at least a given level of ROI?
2. How much more may an automated workcell cost for a desired increase in productivity?
3. How many hours should the automated system work so the payback period is within a certain number of years?

6.1.3 Comparison of Alternatives

EARW can compare up to 10 different workcells at a time. Therefore, up to 10 sets of values related to automated task can be entered by the user. The result of comparison will be shown for a chosen independent variable (as the horizontal axis) and a chosen dependent variable (as the vertical axis). This function of EARW was not used in this study.

6.2 Results of Economic Analysis

6.2.1 Workcell costs versus human inspector labor cost

The cost of robotic workcell was estimated by the retail prices listed in industrial catalogs and through telephone survey. The items which were not standard off- the-shelf products, such as the end-effector, the estimation was based on the components to be used and the cost of development. For the purpose

of comparison, two types of automated inspection workcell were studied: flexible automation and hard automation.

The estimated costs for the components of the inspection workcells are given in Section 4. The total costs of flexible and hard automation inspection workcells were estimated to be $56,500 and $40,000, respectively. These estimated costs were based on the situation when a small number of workcells were purchased. When a large number of workcells are needed, it is conceivable that the real costs could be substantially lower than these estimated costs. During economic analysis, a price range of 25 to 60 thousand dollars was used.

The labor cost for human inspectors was estimated to include an hourly pay of $7.50 per hour and an indirect cost of $5.00 per hour; therefore, a total cost of $12.50 per hour was used as the baseline value.

6.2.2 Equivalent capacity study (automated workcell versus human inspector)

One purpose of the economic analysis is to determine the number of automated workcells required to keep the same plant throughput as the plant with the human inspectors. That is, if the production rate is held constant then the equivalent number of automated workcells depends on the lumped characteristics of the total number of human inspectors. The economic analysis model EARW calculates the number of workcells on the basis of daily productivity, work hours, task inefficiency, and busy ratio. This calculation gives a conservative estimate. If a more realistic estimate is needed, the numerical simulation model should be used.

One cautionary note is that this result, while important to consider, is not complete since defect detection rates are not included as part of this model. On the basis of total throughput, many more workcells are needed, however, when one considers defect detection (as part of complete feasibility) there is not such as dramatic difference.

The number of units produced per hour per man is 394 and the number of units produced per hour per workcell is 234 to yield the daily production of 100,000 units. These statistics are computed from the

following model, derived from our survey of existing plants:

1) Conditions -

daily production rate = 100,000 pouches

daily work period = 10 hours

manual task inefficiency = 1%

manual busy ratio = 87.5%

automated task inefficiency = 4%

2) Equivalent numbers -

human workers = 22

automated workcells = 41

6.2.3 Baseline values for economic analyses

The economic analysis was conducted in three parts. The first two were wide-range feasibility and parametric analyses, one for flexible automation and the other one for hard automation. The third part was conducted to demonstrate, for either type of automated workcell replacing both pre-retort and post-retort inspection workers, the influence of workcell cost, work hours per day, and annual work volume on the simple payback years.

Figures 38 and 39 show the baseline values for engineering economic analysis for flexible automation and hard automation, respectively. These baseline values were derived from our survey of current production facilities. The assumptions for these baseline values are as follows:

1. The inspection time for a robotic workcell is the same for both pre-retort and post-retort inspection lines, that is, we assume that the workcell can do either job in the same time period.

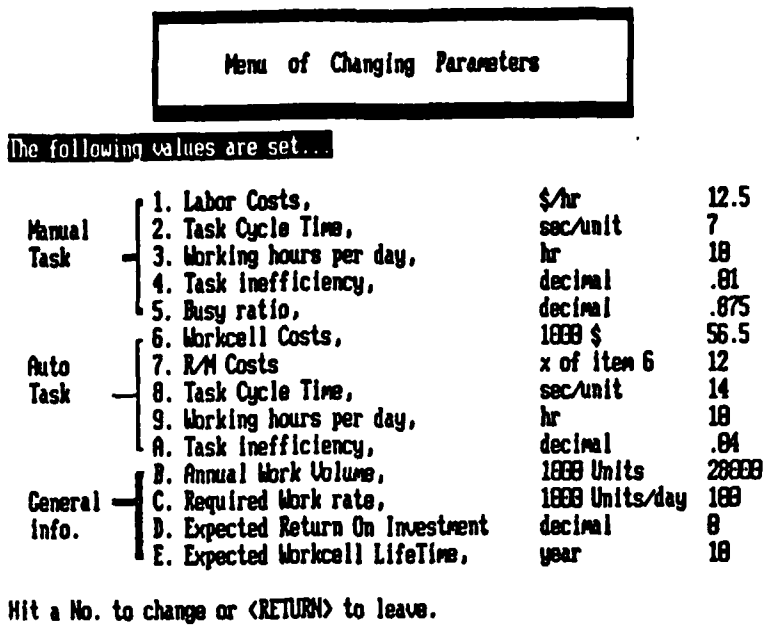


Figure 38: EARW table showing parameters of flexible automation.

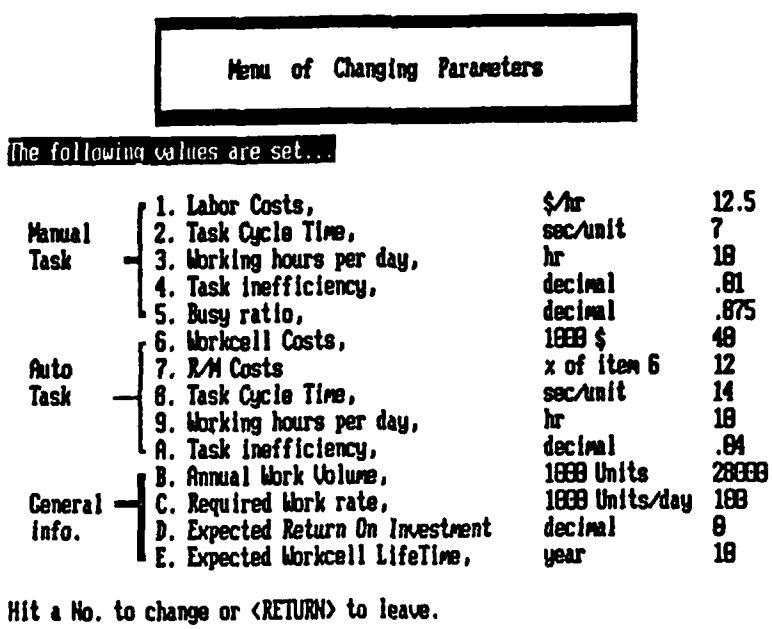


Figure 39: EARW table showing parameters of hard automation.

2. The task inefficiency of a robotic workcell represents the combination of defect detection errors in the image analysis and failures due to pouch mishandling. Although they are lumped together, the majority of the inefficiency is assumed to be due to mishandling, based on our image analysis results in Section 5.

3. The task inefficiency of human inspectors is the percentage of false decision making.

4. The number of working days in one year is 280 days.

5. The lifetime of a workcell is 10 years.

Six cases of economic analysis were conducted. They were:

1. flexible automation workcell (FAW) versus human inspectors for pre-retort lines,

2. hard automation workcell (HAW) versus human inspectors for pre-retort lines,

3. FAW versus human inspectors for post-retort lines,

4. HAW versus human inspectors for post-retort lines,

5. overall analysis of FAW, and

6. overall analysis of HAW.

For each case, we performed both a pessimistic and an optimistic analysis to determine the range of results. For cases 5 and 6, we also performed the baseline case analysis. The pessimistic cases are those with the best scenarios of human inspectors versus the worst scenarios of automated workcell, and the optimistic case were the best scenarios of automated workcell versus the worst scenarios of human inspectors.

6.2.4 Feasibility analysis

It was found that in order to inspect 28 million pouches per year on the pre-retort lines, 13 workcells were required to replace 8 human inspectors if the workcells were operating 16 hours per day. In this pessimistic

case for FAW, the investment would be paid back in 12.104 years. The ROI was less than zero because the payback period is longer than the 10 years project lifetime.

In the optimistic case, for FAW to replace pre-retort inspectors, eight workcells were required to replace 8 inspectors based on a 24 daily working hour schedule for the workcells. The simple payback period was 1.276 years and the ROI for the project (over system life time of 10 years) was 78.127%.

In the pessimistic case for FAW to replace post-retort inspectors, thirteen workcells were required. The simple payback period was 3.546 years and the ROI was 25.251%.

In the optimistic case for FAW versus post-retort inspectors, nine workcells were necessary to replace 15 inspectors. The simple payback period was 0.744 years and the ROI was 134.378%.

For the analagous situations employing HAW for the pre-retort lines, ROI was less than 0 in the pessimistic case. The result of the optimistic case indicated that the simple payback period was 3.156 years and the ROI was 29.251%. For the post-retort inspection lines, the simple payback period was 8.975 years for the pessimistic case, and the ROI was 2.126%. For the optimistic case, the simple payback period was 1.56 years and the ROI was 63.626%.

There are 2 ways to perform economic analysis for the overall inspection lines (considering both pre-retort and post-retort lines together):

1. Use the formula which gives the weighted average of ROIs for pre-retort and post-retort lines to calculate the overall ROI values from the results of separate pre-retort lines analysis and post-retort lines analysis.

2. Directly input the overall task cycle time as 7 seconds (2.5 + 4.5) for human inspectors and 14 seconds (7 + 7) for automated workcell into the program EARW and keep all other values the same.

The result of overall analysis showed that, in the pessimistic case, a workcell using flexible automation will be paid back in 5.485 years and the ROI will be 12.751%. For HAW, the payback period will be 4.042 years and the ROI will be 21.126% in the pessimistic case. It also showed that, in the optimistic case, both

FAW and HAW were going to give us 108.002% of ROI. For the baseline (i.e., average) case, it was found that ROI would be 52.126% using FAW and 60.501% by using HAW.

6.2.5 Parametric analysis

This part of study investigated the dependence of ROI on some important factors, such as labor costs, workcell costs, operation and maintenance costs (R/M costs), workcell task cycle time, and work hours per day.

In one example, it was found that 1) if the workcell cost $50,000, the ROI was greater than 45% if labor costs were higher than $11/hr, and 2) if the labor costs increased by $1/hr, an investment of $5000 more could be justified, maintaining the same ROI. From all the parametric studies performed using the baseline values listed in Figures 38 and 39, the following observations were made:

1) ROI decreased by 2% as R/M costs increased by 1.5% of workcell costs when labor costs were $11/hr.

2) The effect of R/M costs decreased when labor costs increased.

3) ROI and workcell cycle time (WCT) are inversely related and that relationship is nonlinear. Therefore, a small reduction in workcell cycle time could cause a rapid increase in ROI.

4) ROI would increase about 1% if workcells worked one more hour per day.

5) If a workcell cost more than $60,000, workcell cycle time needed to be less than 15 seconds in order to have higher than 15% ROI.

The result of this parametric analysis not only provides us more insights about the feasibility of the automated inspection workcells but also shows which part of the workcell is the most influential on ROI.

Specific cases:

Figure 40 is the result of economic analysis using the baseline values shown in Figure 38. It was estimated that 41 automated workcells were needed to replace 22 human inspectors with comparable productivity (100,000 pouches inspected per day). The simple payback period for this particular case was 3.459 years and the overall return on investment of 41 automated workcells was 26.126%.

Based on the given values :

Annual Work Volume : 28000.00 x 1000 units/year
Number of WORKER available : 22.000
Number of WKCLL available : 41.000

	Manual Task	vs.	Automatic Task	
Given working hours/day	10.00		10.00	hrs/day
Hrly Unit Work Rate	445.500		246.857	units/hr
Daily Unit Work Rate	4455.000		2468.572	units/day
Daily Work Rate	90.810		181.211	x 1000 units/day
number of DAYs required	285.685		276.649	days
Total Labor Cost	785.634			x 1000 $
Total Fixed Cost			2316.500	x 1000 $
Total Operating Cost			115.825	x 1000 $
Expected Payback Year *1			3.459	years
Return On Investment *2			26.126	x

*1: EROI is 0.00 x *2: System LifeTime is 10 years

Press <Enter> to continue..

Figure 40: EARW output for flexible automation system.

It was believed that the $56,500 was a good estimate for the automated workcell designs developed in this research. However, this estimated cost does not consider the possibility of discounted price normally anticipated when ordering a large number of workcells at one time. The effect of workcell cost is shown in combination with annual work volume and workcell work hours in Figures 41 and 42, respectively. For an annual work volume of 28 million pouches inspected over a period of approximately 280 days, reducing the unit workcell price from $56,500 to $26,000 will shorten the simple payback period from 3.1 years to approximately 1.5 years. Some combinations of workcell costs and annual work volumes do not allow the payback to occur within 3 years (upper right corner of Figure 41). However, as shown in the lower left corner of Figure 41, some combinations of workcell costs and work volumes may expect the payback within 2 years.

As expected, one of the most significant factors which affect the return on investment in this project is the number of hours the workcell is operated per day. At $56,500 per unit cost, a payback period of 1.9

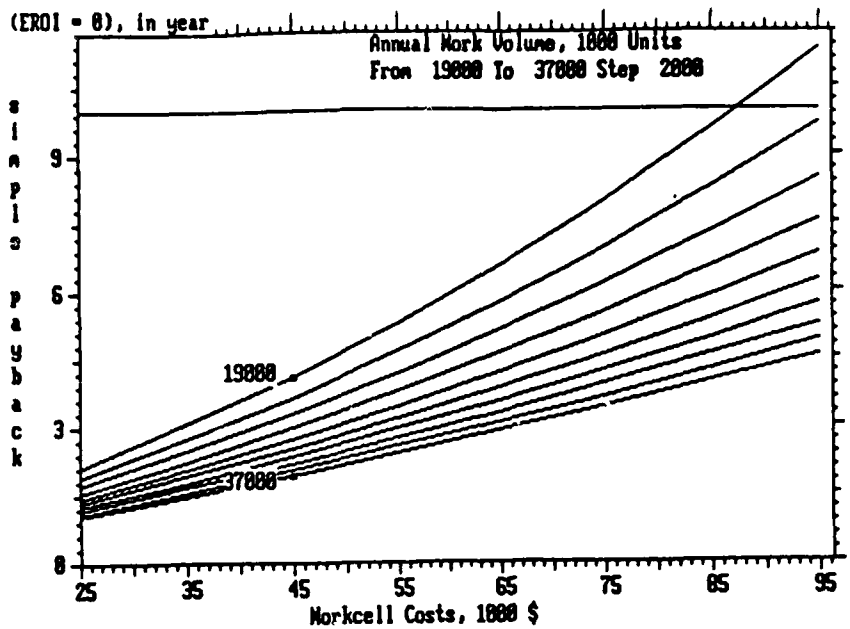


Figure 41: EARW simple payback output for flexible automation system.

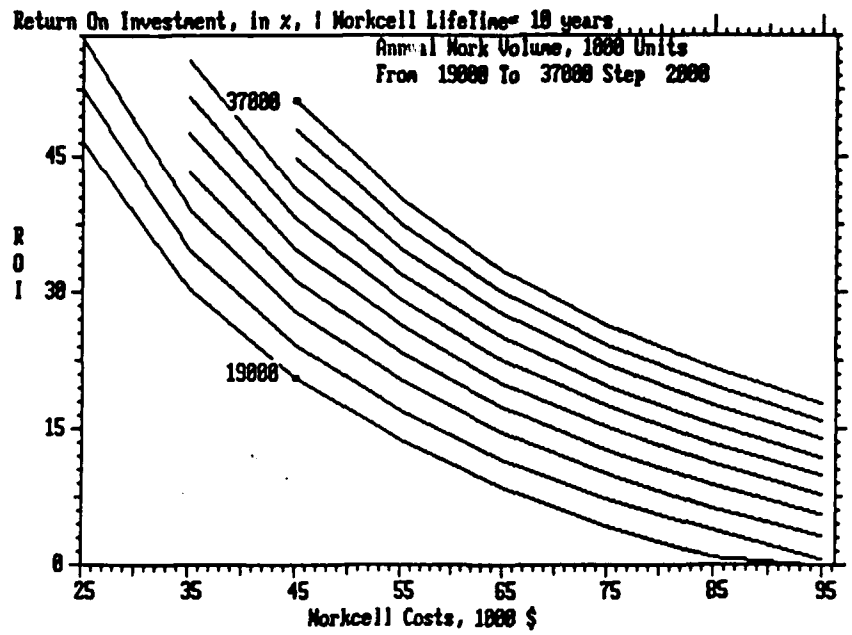


Figure 42: EARW return on investment output for flexible automation system.

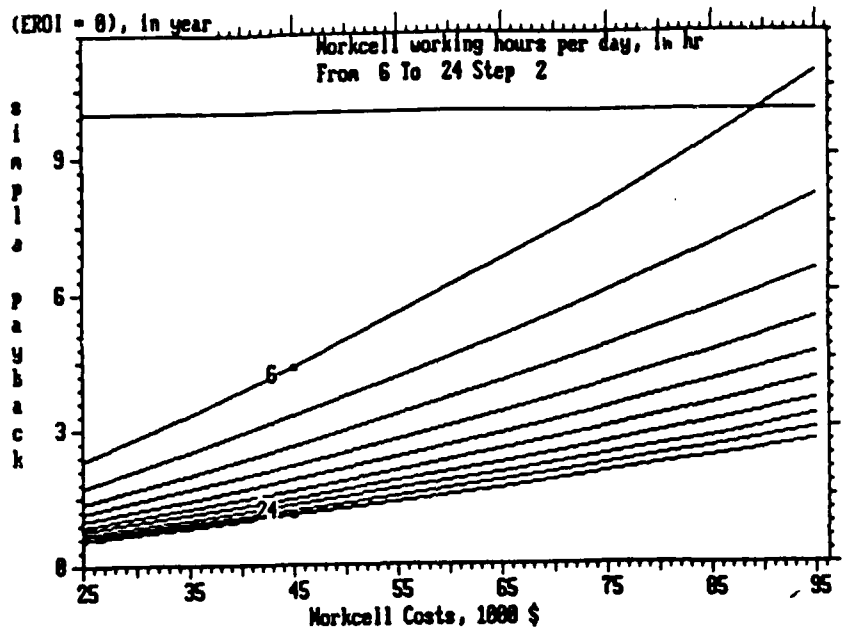


Figure 43: EARW simple payback output controlled by working hours.

years can be expected if the workcell is operated 16 hours per day (Figure 43). This is mainly because the number of workcells required to maintain the productivity level is reduced. With the 16 hour work day schedule, if the workcells can be made available at $26,000 per unit, the simple payback period will become less than 1.5 years.

6.3 Conclusions

The following conclusions are drawn from this study:

1. The numerical models developed in this study can simulate, with satisfactory accuracy, the MRE pouch inspection lines employing either human inspectors or automated workcells.

2. A conceptual design of a robot-based automated MRE pouch inspection workcell as well as a functional end- effector was developed and evaluated. The workcell was found technologically feasible; however, further experimental work is needed to field test the effectiveness of the end-effector.

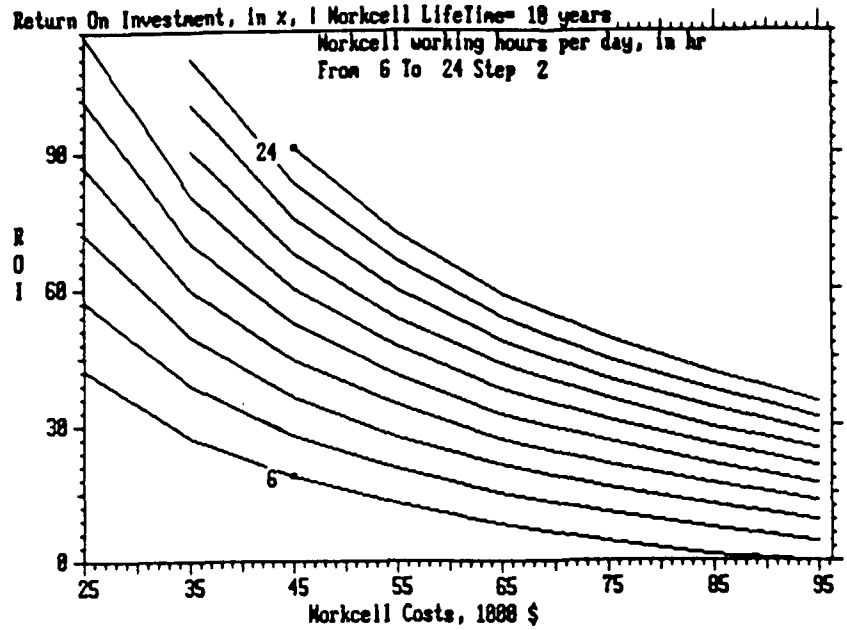


Figure 44: EARW return on investment controlled by working hours.

3. The ranges of design and operating parameters which would provide economic feasibility for automated inspection workcells, including flexible and hard automation, have been identified.

4. Using the operating parameters obtained from an example MRE manufacturer as the baseline values, it is reasonable to expect that the payback on the investment can occur within two years if the workcells operate 10 hours a day and their unit cost is less than $50,000.

   This suggests that a hard automation scheme is preferable to a flexible automation scheme. Figure 45 shows the analysis for hard automation systems priced at $40,000. If the price is reduced as the quantity increases (Figure 46), then the return on investment improves with payback in 1.7 years as shown in Figure 47.

Based on the given values :

Annual Work Volume : 20000.00 x 1000 units/year
Number of WORKER available : 22.000
Number of WKCLL available : 41.000

	Manual Task	vs.	Automatic Task	
Given working hours/day	10.00		10.00	hrs/day
Hrly Unit Work Rate	445.500		246.857	units/hr
Daily Unit Work Rate	4455.000		2468.572	units/day
Daily Work Rate	98.010		181.211	x 1000 units/day
number of DAYs required	285.685		276.649	days
Total Labor Cost	785.634			x 1000 $
Total Fixed Cost			1648.000	x 1000 $
Total Operating Cost			82.000	x 1000 $
Expected Payback Year *1			2.331	years
Return On Investment *2			41.626	x

*1: EROI is 0.00 x *2: System LifeTime is 10 years

Press <Enter> to continue..

Figure 45: EARW output for hard automation solution.

Menu of Changing Parameters

The following values are set...

Manual Task	1.	Labor Costs,	$/hr	12.5
	2.	Task Cycle Time,	sec/unit	7
	3.	Working hours per day,	hr	10
	4.	Task inefficiency,	decimal	.01
	5.	Busy ratio,	decimal	.875
Auto Task	6.	Workcell Costs,	1000 $	30
	7.	R/M Costs	x of item 6	12
	8.	Task Cycle Time,	sec/unit	14
	9.	Working hours per day,	hr	10
	A.	Task inefficiency,	decimal	.04
General info.	B.	Annual Work Volume,	1000 Units	20000
	C.	Required Work rate,	1000 Units/day	100
	D.	Expected Return On Investment	decimal	0
	E.	Expected Workcell LifeTime,	year	10

Hit a No. to change or <RETURN> to leave.

Figure 46: EARW parameters for reduced cost hard automation.

Based on the given values :

Annual Work Volume : 20000.00 x 1000 units/year
Number of WORKER available : 22.000
Number of WKCLL available : 41.000

	Manual Task	vs.	Automatic Task	
Given working hours/day	10.00		10.00	hrs/day
Hrly Unit Work Rate	445.500		246.857	units/hr
Daily Unit Work Rate	4455.000		2468.572	units/day
Daily Work Rate	90.810		181.211	x 1000 units/day
number of DAYs required	285.685		276.649	days
Total Labor Cost	705.634			x 1000 $
Total Fixed Cost			1230.000	x 1000 $
Total Operating Cost			61.500	x 1000 $
Expected Payback Year *1			1.699	years
Return On Investment *2			50.376	x

*1: EROI is 0.00 x *2: System LifeTime is 10 years

Press <Enter> to continue..

Figure 47: EARW output for reduced cost hard automation.

7 Conclusions and Recommendations

The purpose of this short term project STP 11 was to determine the technical and economic feasibility of using machine vision and robotics in packaged food manufacturing. The results of our research suggest that this approach is both technically and economically feasible; a combination of focused illumination of the packages with transport done by fixed automation yields a system with greater than 95% accuracy that can be paid back in 2 years with an ROI of 40%.

7.1 Technical Outcomes

Two simulation tools were developed, namely a numerical simulation and a graphics simulation. For the numerical simulation, an object oriented library was created. Such a library allows the design of numerical simulation for several workcells by simply interconnecting the building blocks (objects). Each piece was an object class written in C++. In this way, the simulation of small and large systems was straightforward.

Information from an existing MRE factory located in Texas was used to create a numerical simulation model. The simulated model was fine-tuned until it matched the average behavior in terms of throughput and error rate. Then, the model was extrapolated to match the average performance of two other companies. A second simulation based on typical robotic workcell parameters was implemented. The topology (i.e., configuration) of the robot-based simulation was manipulated until its performance matched the average of the three companies.

It was established that the overall topology of the inspection plant needs to be changed in order to use robotic inspection workcells. The reason for this is to increase the number of parallel inspection lines to compensate for (individually) slower automatic inspectors. If detection task accuracy is maintained (and improved) by using machine inspection systems, then product throughput can be maintained even with workcells that are individually slower.

Even though, at this point of the research, there are not sufficient known parameters to build a complete simulation model, the simulation tool developed here has proved to be of great aid. Once a working,

dynamic prototype of the complete workcell is available, it will be possible to create an accurate simulation model with data gathered from experimentation. Such a simulation can then be extrapolated to regions (of the parameter space) were it is difficult or impossible to take the real prototype. This close-couple, prototype and its simulation, can be used to design an optimal robotic and machine vision-based workcell.

A second simulation tool created was an interactive graphics simulation. The user was allowed to "fly" through the graphics model to visualize all its 3D aspects. A library of graphically modeled objects was created so that the overall workcell was easily built. A human-based and a robot-based workcells were simulated.

By interacting with these graphics models it was clear that robot-based workcells are larger in size and require a completely new layout distribution. Some of the conclusions of the numerical simulation were reproduced in the graphics simulation. For instance, two parallel inspection lines were used for the robotic workcell while only one was used for the human workcell. This was done in order to handle the same number of pouches per second.

The flexibility given by graphics and numerical simulation represents a powerful tool for future implementation. The models presented in this report can be considered as demonstrative ones. However, the same tools developed and used by this research can also be used in the future when the real implementation will be attempted.

Real (non-simulated) models of robot manipulators were discussed. The two limit cases of automation –flexible and fixed– were presented. Out of all possible implementations two particular hardware models were discussed. One based on industrial robot arms (flexible-automation) and one on customized designed devices (hard-automation).

It was established that despite of the generality and easy implementation of flexible-automation it can be very expensive. On the other hand, the fixed-automated version presented a complex implementation since designing and prototyping must be performed before any device is integrated in a plant.

From this analysis it was clear that a balance between fixed-automation and flexible-automation must

be obtained. In this way it will be possible to design an optimal automatic workcell takes advantage of both methods.

Making use of currently available machinery can decrease the cost and complexity of the workcell implementation. For instance, the problem of precise grasping (known orientation and position) can be simplified if the robot systems are tightly coupled with the automatic machine used to process the pouches (currently used filler-machine). In this case the robot system can make use of whatever mechanisms the filler-machine uses to hold steady the empty pouch, fill it and seal it. This particular solution requires further analysis of the currently used machinery. Such analysis is recommended to be done in the future.

Finally the most critical component of the MRE inspection workcell was analyzed –the machine vision system–. It was established that in order to detect the broad spectrum of the pouch's defects, several different techniques are required.

A detection model for defects that exposes the pouch's metal coating was discussed. Theoretical analysis showed why the model works and prototyping confirmed it.

The conclusion of the testing of the system prototype was that the error-rate must decrease before it is applicable. However, several techniques (based on the theoretical analysis) to improve the system were presented.

There are still many other vision techniques that have not yet been explored. It may be interesting to analyze and perhaps experiment with some of them. For example, the use of other intervals of the electromagnetic spectrum. The electromagnetic behavior of metal coating as opposed to plastic or paint coating could facilitate the use of other electromagnetic field wave lengths. One can speculate and talk about a micro wave based scanner (radar like) to located metal coating exposure (abrasion, delamination, etc) or anomalies (cut, holes, wrinkles, etc). Nevertheless, such research is beyond the scope of this feasibility study.

Polarized light can be used to increase the defect detection probability. It is known that for every material there exists an angle θ_p called the *polarization angle* such that all the reflected light (especially

specular-reflected) at that angle is linearly polarize (in a plane parallel to the reflector surface). In 1812 Sir David Brewster established that that:

$$\theta_p = n_1/n_2$$

where n_1 is the index of refraction of the air and n_2 is the index of refraction of the surface material. This phenomenon could, in principle, be added to the multiple-illumination and color-subtractive technique to remove in a more efficient way the reflection due to defect-free surfaces. This will be plausible if the refraction index of the metal coating is different from that of the paint or plastic coating, so that, the two materials will present a different *polarization angle.* In this way it is possible, by placing a linear polarizer in front of the camera and light source (at the correct polarization angle), to remove almost completely one of the material reflections (ideally the one from the non-metal material).

Even though it may be impossible to remove completely this component, the system will decrease it, improving in this way the error detection rate. Such a system was tested successfully for the purpose of this research in a few pouch samples, however, so many new parameters were introduced that a full development of a first optimal prototype would require several months of analysis and experimentation. The investigators consider this technique to represent a good follow up to the technique described in section five.

More technical data on the MRE pouch's material must be gathered before new techniques are attempted. For instance, exact material composition and its interaction with electromagnetic radiation. It is important for the development of analytical models and for efficient prototyping to have a full description of the material that composes the pouch. For instance, in order to determine the optimal color for the filter used by the color-subtractive technique, it is important to know the correct experimental frequency spectrum of the "green" coating. In order to determine the correct field of illumination to be used (bright-field or dark-field model) combined with polarization light techniques, it is important to know the refraction index of the "green" coating and the metal coating. All this information can be obtained if the pouch is analyzed in a materials laboratory.

New techniques must be developed so that it is possible to deal with other types of defects. Tactile sensors could be used to scan the seal's surface looking for wrinkles, material entrapped on the seal, defective tear notches, etc. A two-finger like robot end-effector could easily slide over the seal while reading the thickness of it. Wrinkles and entrapped material are definitively notable disturbances of the seal thickness. The same device can be used to detect the position and size of the tear notch. Such a device can be integrated together with the transport system so that no extra time is added to the inspection process.

Once the different components have been optimally determined it will be of great importance ple-ment a complete robotic workcell. This will allow to determine overall behavior of the system. Experimentation on the real prototype will determine the critical points to be strengthened. Experiments will also determine the real characteristics of the workcell. Such characteristics can be used to feed the numerical and graphical simulator. Simulators can be then used again to model the complete plant so that a more realistic topology and layout of a plant can be determined.

7.2 Recommendations

In summary, our theoretical work, simulations, experiments and economic analysis suggest that automated inspection of MRE pouches is technically and economically feasible. Technical feasibility is based on fundamental principles of the interaction of tuned radiation (colored light) and the surface under test. Our detection results would not be achievable by image processing alone, but by paying attention to physical characteristics, one achieve both feasibility and efficiency. Coupling the image analysis with fixed automation for product transport will make the system amenable to use in a production environment.

Our recommendations for the next step are:

1. The problem of MRE pouch placement and transport was relatively easy to solve in simulation, but a prototype needs to be built to conduct experiments to verify that the rates predicted by the simulation are achievable.

2. It was established that the detection system is feasible, but needs to be improved. There are no false

positive defects detected, but some (5%) of the true defects went undetected. These defects were on pouch locations of high surface curvature; the loss is likely due to shading introduced. Polarizing the colored light is an exciting possibility for fine tuning the system to avoid losing these kinds of defects and should be studied in the next phase.

3. Not only should the accuracy of the system be studied, but the inspection time can be improved. The hardware used for the timing in this report is vintage 1990; even in the last 4 years there have been such dramatic changes in the kinds of special purpose computational hardware available that the 7 second figure, although a baseline is very pessimistic by todays standards. We recommend that a prototype be constructed from those special purpose image processing boards available now (1994), rather than the specifications used herein, although even by the older standards the underlying approach is economically feasible.

References

[1] Asada Haruhiko and Kakumoto Yoshiki. The dynamic rcc hand for high-speed assembly. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 120–125, Philadelphia, Pennsylvania, April 1988.

[2] AT&T. Bell Laboratories, USA. *UNIX User's Manual*, 1979.

[3] Basta Robert A., Mehrotra Rajiv and Varanasi Murali R. Collision detection for planning collision-free motion of two robot arms. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 638–640, Philadelphia, Pennsylvania, April 1988.

[4] Beltrame F., Dario P., Fadda M., Marcacci M., Marcenaro G., Martelli S. and Visani A. A laboratory for computer-assisted orthopedic surgery. In *Proceedings of Fifth International Conference on Advanced Robotics*, pages 882–887, Pisa, Italy, June 1991.

[5] Boyer Martin, Daneshmend Laeeque K., Hayward Vincent and Foisy Andre. An object oriented paradigm for the design and implementation of robot planning and programming systems. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 294–209, Sacramento, California, April 1991.

[6] Brennemann A. E., Hollis R. L. and Musits B. L. Sensors for robotic assembly. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1606–1610, Philadelphia, Pennsylvania, April 1988.

[7] Brett P.N., Sacklock A.P. and Khodabandehloo K. Research towards generalized robotic systems for handling non-rigid products. In *Proceedings of Fifth International Conference on Advanced Robotics*, pages 1530–1533, Pisa, Italy, June 1991.

[8] Bui-Tuong Phong. Illumination for computer generated pictures. *Communication of the Association for Computer Machinery*, 6(18):311–317, June 1975.

[9] Burdea G. *Robot Assembly of Jigsaw Puzzle Pieces.* PhD thesis, New York University, New York, Octuber 1987.

[10] Burdea G. and Speeter T. Portable dextrous force feedback master for robotics telemanipulation (p.d.m.f.f). In *Proceedings NASA Conference on Space Telerobotics.*, volume 2, pages 153–161, Pasaden, CA, January 1989.

[11] Burdea G and Zhuang J. Dextrous telerobotics with force feedback - an overview - part 2: Control and implementation. *Robotica, UK*, 9:291–298, 1991.

[12] Burdea G. and Zhuang J. Dextrous telerobotics with force feedback - an overview. part 1: Human factors. *Robotica, UK*, 9:171–178, 1991.

[13] Chang Shaw Jen, DiCesare Frank and Goldbogen Geoffrey. Evaluation of diagnosability of failure knowledge in manufacturing systems. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 695–701, Cincinnati, Ohio, May 1990.

[14] Chen Philip C.L. Time lower bound for manufacturing aggregate scheduling problem. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 830–835, Sacramento, California, April 1991.

[15] CIS Graphics, Inc., Westford, Massachusetts. *Dimension 6 User's Manual*, 1988.

[16] Clark J. Peter. *Biotechnology and Food Process*, chapter 13, pages 405–413. Marcel Dekker, Inc., New York and Basel, 1990.

[17] Cox Ingemar J. C++ language support for guaranteed initialization, safe termination and error recovery in robotics. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 641–643, Philadelphia, Pennsylvania, April 1988.

[18] Cronshaw T. The "intelligent" chocolate packing robot. *Proceedings of International Symposium on Industrial Robotics*, pages 151–158, October 1990.

[19] Dalum L. and Lund T. Trimming robot for fish fillets. In *Proceedings of Symposium on Industrial Robots*, pages 145–149, October 1990.

[20] Dario Antonelli and Marchis Vittori. On the dexterity of machines: Some (historical) considerations between system modeling and industrial applications. In *Proceedings of Fifth International Conference on Advanced Robotics*, volume 1, pages 504–508, Pisa, Italy, June 1991.

[21] Davies B.L., Hibberd R.D., Ng W.S., Timoney A.G. and Wickham J.E.A. A surgeon robot for prostatectomies. In *Proceedings of Fifth International Conference on Advanced Robotics*, pages 871–875, Pisa, Italy, June 1991.

[22] Drake James D., Joy Michael, Goldenberg Andrew and Kreindler David. Computer and robot assisted resection of brain tumors. In *Proceedings of Fifth International Conference on Advanced Robotics*, pages 888–892, Pisa, Italy, June 1991.

[23] Eckel Bruce. *Using C++*. Osborne Mc Graw-Hill, California, 1989.

[24] Erdmann Michael, Mason Matthew T. and Vanecek George. Mechanical part orientation: the case of a polyhedron on a table. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 360–365, Sacramento California, April 1991.

[25] Fang W., Ting K.C. and Giacomelli G. A. Engineering economics of scara robot-based plug transplanting workcell. In *American Society of Agricultural Engineering*, St. Joseph, MI, 1991.

[26] Frost A.R. Robotic milking: a review. *Robotica, UK*, 8:311–318, 1990.

[27] Gagliardi G., Hatch G.F. and Sarkar N. Machine vision applications in food industry. In *Proceedings of SME VISION'86 Conference*, Detroit, Michigan, June 1986.

[28] Goldberg Kenneth Y. and Mason Matthew T. Bayesian grasping. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1264–1269, Cincinnati, Ohio, May 1990.

[29] Goldberg Kenneth Y., Mason Matthew T. and A. Erdmann Michael. Generating stochastic plans for a programmable parts feeder. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 352–359, Sacramento, California, April 1991.

[30] Graham James H., Alexander Suraj M. and Lee Won Young. Object-oriented software for diagnosis of manufacturing systems. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1966–1971, Sacramento, California, April 1991.

[31] Gurnani Heresh, Anupindi Ravi and Akella Ram. Control of batch processing system. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1772–1777, Sacramento, California, April 1991.

[32] Hackwood S. and Beni G. Sensor and high-precision robot research. In *Proceedings of Robotics Research: The first International Symposium*, pages 529–545, Cambridge, Massachusetts, 1985.

[33] Harrel R.C., Adsit P.D., Munilla R.D. and Slaughter D.C. Robotic picking of citrus. *Robotica, UK*, 8:269–278, 1990.

[34] Hewlett Packard, USA. *Starbase Graphics Techniques: HP-UX Concepts and Tutorials*, 1988.

[35] Hirose Shigeo and Ma Shugen. Moray drive for multijoint manipulator. In *Proceedings of Fifth International Conference on Advanced Robotics*, pages 521–525, Pisa, Italy, June 1991.

[36] Hormman Andreas and Rembold Ulrich. Development of an advanced robot for autonomous assembly. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2452–2457, Sacramento, California, April 1991.

[37] Horn Berthold Klaus Paul. *Robot Vision.* The MIT press, McGraw-Hill Company, Cambridge, Massachuset, 1991.

[38] Jacobsen S.C. Design of the Utah/MIT Dextrous Hand. In *Proceedings of IEEE International Conference in. Robotics and Automation*, pages 1520–1232, San Francisco, CA, April 1986.

[39] Kabuka M., Glaskowski P. and Miranda J. A flexible high performance robot arm controller. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 630–632, Philadelphia, Pennsylvania, April 1988.

[40] Kassler Michael. Introduction to the special issue on robotics and food-handling. *Robotica, UK*, 8:267–268, 1990.

[41] Kessler M. Robotics and prawn-handling. *Robotica, UK*, 8:299–301, 1990.

[42] Khodabandehloo K. Robotic handling and packaging of poultry products. *Robotica, UK*, 8:285–297, 1990.

[43] King F. G., Puskorius G. V. and Yuan F. Vision guided robots for automated assembly. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1611–1616, Philadelphia, Pennsylvania, April 1988.

[44] Lee Jay, Bastuscheck Marc and Little Arthur D. Intelligent sensing and machine reasoning for robotic manipulation of irregular moving objects. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1562–1565, Sacramento, California, April 1991.

[45] Magnani G., Bologna P. and Lovati G. Modelling and simulation of an industrial robot. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1532–1538, Philadelphia, Pennsylvania, April 1988.

[46] Marbot Pierre-Henrry and Hannaford Blake. Mini direct drive robot arm for biomedical application. In *Proceedings of Fifth International Conference on Advanced Robotics*, pages 859–864, Pisa, Italy, June 1991.

[47] Mason M.T. *Manipulator Grasping and Pushing Operations.* PhD thesis, Massachusetts Institute of Technology, Boston, 1982.

[48] Menga G. and Mancin M. A framework for object oriented design and prototyping of manufacturing systems. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 128–133, Sacramento, California, April 1991.

[49] Miller David J. and Charleene Lennox R. An object-oriented environment for robot system architecture. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 352–361, Cincinnati, Ohio, May 1990.

[50] Montana David J. The condition for contact grasp stability. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 412–417, Sacramento California, April 1991.

[51] Montermelo Malvin D. Nasa's automation and robotics technology development program. *IEEE, U.S. Government Work*, 2:977–986, 1986.

[52] Owens Thomas A. and Luh Peter B. A job completion time estimation method for work center scheduling. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 110–115, Sacrament, California, April 1991.

[53] Park T. H. and Lee B. H. An approach to robot motion analysis planning for conveyor tracking. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 70–75, Sacrament, California, April 1991.

[54] Purnell G., Maddock N.A. and Khodabandehloo K. Robot deboning for beef forequarters. *Robotica, UK*, pages 303–310, 1990.

[55] Schoenwald J.S., Beckham M., Rattner R.A., Vanderlip B. and Shi B.E. End effector actuation with a solid state motor. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 108–113, Philadelphia, Pennsylvania, April 1988.

[56] Shannon Robert E. *System Simulation: the art and the science.* Prentice Hall, New Jersey, 1975.

[57] Sharon Andre, Hogan Neville and Hardt David E. High bandwidth force regulation and inertia reduction using a macro/micro manipulator. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 126–132, Philadelphia, Pennsylvania, April 1988.

[58] Shin Kang G. and Zheng Qin. Scheduling job operations in an automatic assembly line. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 176–181, Cincinnati, Ohio, May 1990.

[59] Spreng Michael. Dealing with unexpected situations during execution of robot motions. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 64–69, Sacramento, California, April 1991.

[60] Stephen C. Dewhurst Kathy T. Stark. *Programming in C++*. Prentice Hall, New Jersey, 1989.

[61] Sturzenbecker Martin C. Building an object-oriented environment for distributed manufacturing software. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1972–1978, Sacramento, California, April 1991.

[62] Tarabanis Konstantinos, Tsai Roger Y. and Allen Peter K. Automated sensor planning for robotic vision tasks. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 76–81, Sacramenst, California, April 1991.

[63] Taylor Russell H., Paul Howard A., Kazanzides Peter, Mittelstadt Brent D., Hanson Willian, Zuhars Joel, Willson Bill, Musits Bela, Glassman Edward and Bargar Willian L. Taming the bull: Safety in a precise surgical robot. In *Proceedings of Fifth International Conference on Advanced Robotics*, pages 865–870, Pisa, Italy, June 1991.

[64] Tedford J.D. Development in robot grippers for soft fruit packing in New Zeland. *Robotica, UK*, 8:279–283, 1990.

[65] Ting K.C. and Wang Da-Lee. Stp11 internal technical report: Economical analysis. Technical report, CRAMTD, 1992.

[66] Troncy A., Martinez M. T., Bara S. El and Hugues C. Modular robots - graphical interactive programming. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1739–1742, Philadelphia, Pennsylvania, April 1988.

[67] Ulrich Nathan and Kumar Vijay. Mechanical design methods of improving manipulator performance. In *Proceedings of Fifth International Conference on Advanced Robotics*, pages 515–520, Pisa, Italy, June 1991.

[68] van Dam S.K. Feiner J.F. Hughes J.D. Foley A. *Computer Graphics: Principles and Practice*. Addison-Wesley, New York, 1990.

[69] Verghese Gilbert, Gale Karey Lynch and Dyer Charle R. Real time motion tracking of three-dimensional objects. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1998–2003, Cincinnati, Ohio, May 1990.

[70] Wang Fei-Yue and Saridis George. Coordination structures for specification of integration in intelligent machines. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2690–2694, Sacramento, California, April 1991.

[71] Warnock J.A. A hidden-surface algorithm for computer generated half-tone pictures. *Technical Report TR 4-15, NTIS AD-753 671, Computer Science Department Univer ity of Utha*, June 1996.

[72] Xie Xuanli Lisa and Espiau Bernard. Clustering validity based image recognition segmentation for IC wafer defects recognition. In *Proceedings of Fifth International Conference on Advanced Robotics*, pages 1404–1409, Pisa, Italy, June 1991.

A Simulation Code

The following list is the code for the complete pre-retort and post-retort models of the Texas plant using SOL.

```
// Default ****************************************************
#include <stdio.h>
#include <stream.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>
// STEP 1 *****************************************************
#include "sol.h"
#include "bclasses.h"
#include "sim.h"

main()
{

  // STEP 2 *****************************************************
  Source       source(3)[4];       //rate 1 pouch/sec
  Conveyor     convey_1(6)[4];     //cap. 6 pouches.
  Inspector    inspector(1)[8];    //insp. time 1 pouch/sec
  Conveyor     convey_2(6)[4];     // cap. 6 pouches
  Transfer     transfer(1,.1)[4];  //cap.1 item, speed. 1sec/item
  Buffer       buffer_A(15)[4];    //cap. 15 pouches.
  Buffer       buffer_B(15)[4];    //cap. 15 pouches.
  Transfer     transf_A(1,.1)[4];  //capacity 1 item, sped: 3sec/item
  Transfer     transf_B(1,.1)[4];  //capacity 1 item, sped: 3sec/item
  Accepted     accepted;           //None.
  Disposed     disposed;           //None.

  // STEP 3 *****************************************************
  install( &source[0],    10 );install( &source[1],    11 );
  install( &source[2],    12 );install( &source[3],    13 );
  install( &convey_1[0],  40 );install( &convey_1[1],  41 );
  install( &convey_1[2],  42 );install( &convey_1[3],  43 );
  install( &inspector[0], 50 );install( &inspector[1], 51 );
  install( &inspector[2], 52 );install( &inspector[3], 53 );
  install( &inspector[4], 54 );install( &inspector[5], 55 );
  install( &inspector[6], 56 );install( &inspector[7], 57 );
  install( &convey_2[0],  401);install( &convey_2[1],  411);
  install( &convey_2[2],  421);install( &convey_2[3],  431);
  install( &transfer[0],  30 );install( &transfer[1],  31 );
  install( &transfer[2],  32 );install( &transfer[3],  33 );
  install( &buffer_A[0],  20 );install( &buffer_A[1],  21 );
  install( &buffer_A[2],  22 );install( &buffer_A[3],  23 );
  install( &buffer_B[0],  24 );install( &buffer_B[1],  25 );
  install( &buffer_B[2],  26 );install( &buffer_B[3],  27 );
  install( &transf_A[0],  301);install( &transf_A[1],  311);
  install( &transf_A[2],  321);install( &transf_A[3],  331);
```

```
    install( &transf_B[0],  302);install( &transf_B[1],  312);
    install( &transf_B[2],  322);install( &transf_B[3],  332);
    install( &accepted,  60 );   install( &disposed,  70 );

    // STEP 4 ************************************************
    // LINE 1 //    // LINE 2 //
    connect(10, 40, UNDEFINED); connect(11, 41,UNDEFINED);
    connect(40, 50, UNDEFINED); connect(41, 52,UNDEFINED);
    connect(40, 51, UNDEFINED); connect(41, 53,UNDEFINED);
    connect(50, 401,UNDEFINED); connect(52,411,UNDEFINED);
    connect(51, 401,UNDEFINED); connect(53,411,UNDEFINED);
    connect(401,30, UNDEFINED); connect(411,31,UNDEFINED);
    connect(30, 20, GOOD);      connect(31, 21,GOOD);
    connect(30, 24, BAD);       connect(31, 25,BAD);
    connect(20, 301,UNDEFINED); connect(21,311,UNDEFINED);
    connect(24, 302,UNDEFINED); connect(25,312,UNDEFINED);
    connect(301,60, UNDEFINED); connect(311,60,UNDEFINED);
    connect(302,70, UNDEFINED); connect(312,70,UNDEFINED);

    // LINE 3 //  // LINE 4 //
    connect(12, 42, UNDEFINED); connect(13, 43,UNDEFINED);
    connect(42, 54, UNDEFINED); connect(43, 56,UNDEFINED);
    connect(42, 55, UNDEFINED); connect(43, 57,UNDEFINED);
    connect(54, 421,UNDEFINED); connect(56,431,UNDEFINED);
    connect(55, 421,UNDEFINED); connect(57,431,UNDEFINED);
    connect(421,32, UNDEFINED); connect(431,33,UNDEFINED);
    connect(32, 22, GOOD);      connect(33, 23,GOOD);
    connect(32, 26, BAD);       connect(33, 27,BAD);
    connect(22,321, UNDEFINED); connect(23,331,UNDEFINED);
    connect(26,322, UNDEFINED); connect(27,332,UNDEFINED);
    connect(321,60, UNDEFINED); connect(331,60,UNDEFINED);
    connect(322,70, UNDEFINED); connect(332,70,UNDEFINED);

    // STEP 5 and 6 ******************************************
    defects(2);
    TFR[0] = 0.1;
    TFR[1] = 0.1;
    TFR[2] = 0.1;
    simulate( 3600.0, .2, FALSE );
}
```

The following is a selection of the SOL software.

```
//*****************************************************************//
//*                                                               *//
//*                  RUTGERS UNIVERSITY                           *//
//*            MAN-MACHINE INTERFACE LABORATORY                   *//
//*                                                               *//
//*                        STP11                                  *//
//*                     SOL LIBRARY                               *//
//*               Basic Classes Definition                        *//
//*                                                               *//
//* bclasses.h           Spring 1992                              *//
//*****************************************************************//

//---------------------------------------------------------------//
//---                         ITEM                            ---//
//---------------------------------------------------------------//

class Item
{
 protected:
  long int ID;
  int state;
  double defect[ DEFECTS_PER_ITEM ];  //The actual quality.
  double estimate[ DEFECTS_PER_ITEM ]; //Given by an inspector.

 public:
  Item()
    {
      state = UNDEFINED;
    }
  void    set_state(int st){ state = st; }
  void    set_ID(int number){ ID = number; };
  void    set_defects(double *def_prob);
  int     show_state(){return(state);}
  int     show_ID(){ return(ID); };
  double  *show_defects(){ return(defect); };
  double  show_defect(int j){return(defect[j]);}
  void    genere_defects();
};

//---------------------------------------------------------------//
//---                       UNIT CLASS                        ---//
//---------------------------------------------------------------//

class Unit
{
 protected:
  int ID;
  int state;
  int type_unit;
  long int item_input;
  long int item_output;
```

```
  Item item;

 public:
  Unit()
    {
      ID = 0;
      state = IDLE;
      item_input = item_output = 0;
    }
  virtual int set_ID(int id ){ ID = id; return( ID );}
  int show_ID(){ return( ID );}
  int set_state( int st ){ state = st; return( state );}
  int show_state(){ return( state ); }
  int show_type(){ return( type_unit );}
  virtual long int show_index(){ return(item_input - item_output);}
  long int show_itmout(){ return( item_output );}
  long int show_itminp(){ return( item_input );}
  virtual int show_item_state(){ return( item.show_state());}

};

//-------------------------------------------------------------//
//---                      SOURCE CLASS                     ---//
//-------------------------------------------------------------//
class Source : public Unit
{
 protected:
  double issue_last;
  double issue_per;
  double issue_delta;

 public:
  Source()
    {
      type_unit = SOURCE_TYPE;
      issue_last = 0;
      issue_per  = 1/SOURCE_PERIOD;
    }
  Source(double rate)
    {
      type_unit = SOURCE_TYPE;
      issue_last = 0;
      issue_per  = 1.0/rate;
    }
  int set_ID(int id )
    {
      ID = id;
      cout << "Source installed. \tID = "<< ID << "\n";
      return(ID);
    }
  long int show_index(){ return(item_output);};
  put_item(Item *out_item);
  double show_per(){return(issue_per);}
};
```

```
//-------------------------------------------------------------//
//---                      BUFFER CLASS                       ---//
//-------------------------------------------------------------//
class Buffer : public Unit
{
 protected:
  int in_stack_ptr;
  int out_stack_ptr;
  int capacity;
  Item stack_item[MAX_BUFFER_CAPACITY];

 public:
  Buffer()
    {
      type_unit = BUFFER_TYPE;
      state = EMPTY;
      in_stack_ptr = 0;
      out_stack_ptr = 0;
      capacity = BUFFER_CAPACITY-1;
    }
  Buffer(int cap)
    {
      type_unit = BUFFER_TYPE;
      state = EMPTY;
      in_stack_ptr = 0;
      out_stack_ptr = 0;
      capacity = cap-1;
    }
  int set_ID(int id )
    {
      ID = id;
      cout << "Buffer installed. \tID = "<< ID << "\n";
      return(ID);
    }
  int show_in_ptr(){ return(in_stack_ptr);}
  int show_out_ptr(){ return(out_stack_ptr);}
  int get_item( Item in_item );
  int put_item( Item *out_item);
  int show_item_state();
};

//-------------------------------------------------------------//
//---                     TRANSFER CLASS                      ---//
//-------------------------------------------------------------//
class Transfer : public Unit
{
 protected:
  double last_transfer;
  double delta_time;
  double current_transfer_time;

 public:
  Transfer()
```

```
    {
      type_unit  = TRANSF_TYPE;
      state = EMPTY;
      last_transfer = 0.0;
      current_transfer_time = TRANSFER_TIME; //4.0 * RAND + 2.0;
    };
  Transfer(int cap, double trans_time)
    {
      type_unit  = TRANSF_TYPE;
      state = EMPTY;
      last_transfer = 0.0;
      current_transfer_time = trans_time;
    };
  int set_ID(int id )
    {
      ID = id;
      cout << "Transfer installed. \tID = "<< ID << "\n";
      return(ID);
    }
  int get_item( Item in_item );
  int put_item( Item *out_item);
};

//-----------------------------------------------------------------//
//---                        CONVEYOR CLASS                     ---//
//-----------------------------------------------------------------//
class Conveyor : public Buffer
{
 public:
  Conveyor()
    {
      capacity = CONVEYOR_CAPACITY-1;
      type_unit = CONVEY_TYPE;
      state = EMPTY;
    }
  Conveyor(int cap)
    {
      capacity = cap-1;
      type_unit = CONVEY_TYPE;
      state = EMPTY;
    }
  int set_ID(int id )
    {
      ID = id;
      cout << "Conveyor installed. \tID = "<< ID << "\n";
      return(ID);
    }
};

//-----------------------------------------------------------------//
//---                        INSPECTOR CLASS                    ---//
//-----------------------------------------------------------------//
class Inspector : public Unit
{
```

```
 public:

  Inspector()
    {
      type_unit = INSPEC_TYPE;
      state = EMPTY;
    }

  Inspector(int insp_time)
    {
      type_unit = INSPEC_TYPE;
      state = EMPTY;
    }

  int set_ID(int id )
    {
      ID = id;
      cout << "Inspector installed. \tID = "<< ID << "\n";
      return(ID);
    }

  int get_item( Item in_item );
  int put_item( Item *out_item);
  int inspection();
};

//-----------------------------------------------------------------//
//---                        ACCEPTED  CLASS                    ---//
//-----------------------------------------------------------------//
class Accepted : public Unit
{
 public:
  Accepted()
    {
      type_unit = ACCEPT_TYPE;
      state = READY;
    }
  int set_ID(int id )
    {
      ID = id;
      cout << "Accepted installed. \tID = "<< ID << "\n";
      return(ID);
    }
  int get_item( Item in_item );
};

//-----------------------------------------------------------------//
//---                      DISPOSED  CLASS                      ---//
//-----------------------------------------------------------------//
class Disposed : public Unit
{
 public:
  Disposed()
    {
```

```
      type_unit = DISPOS_TYPE;
      state = READY;
    }
  int set_ID(int id )
    {
      ID = id;
      cout << "Disposed installed. \tID = "<< ID << "\n";
      return(ID);
    }
  int get_item( Item in_item );
  long int show_index(){ return(item_input);};
};

//-----------------------------------------------------------------//
//---                      PROCESS  CLASS                       ---//
//-----------------------------------------------------------------//
class Process : public Unit
{
 public:
  Process()
    {
      type_unit = PROCES_TYPE;
      state = READY;
    }
  int set_ID(int id )
    {
      ID = id;
      cout << "Process installed. \tID = "<< ID << "\n";
      return(ID);
    }
  int get_item( Item in_item );
  int put_item( Item *out_item );
};
```